Computational Construction and Classification of Some Mutually Orthogonal Latin Squares

By

Amadou Keita

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Mathematics and Statistics
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy in Mathematics and Statistics
at the University of Windsor

Windsor, Ontario, Canada

2026

© 2026 Amadou Keita

Computational Construction and Classification of Some Mutually Orthogonal Latin Squares

By

Amadou Keita

Date of final oral defence: April 10, 2026

The document has received final approval by the Doctoral Committee:

PhD External Examiner: Ian Wanless, Monash University

Co-Supervisor: Curtis Bright

Co-Supervisor: Ilya Shapiro

Program Reader: Dennis Borisov

Program Reader: Mehdi Monfared

Outside Program Reader: Muhammad Asaduzzaman

DECLARATION OF CO-AUTHORSHIP/PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this dissertation incorporates material that is the result of joint research, as follows:

In Chapters 3 and 4 of this dissertation, the findings are the result of collaborative work with Curtis Bright and Brett Stevens as co-authors. The initial conceptual framework for the proposed methodology for Chapter 3 originated from Brett Stevens and the initial conceptual framework for the proposed methodology for Chapter 4 originated from Curtis Bright. Additionally, all aspects of the research, including implementation, were accomplished through joint efforts and collaboration. The initial implementations were run and written by myself and they were later rerun and optimized by Curtis Bright to improve their performance.

The initial conceptual framework for the proposed methodology for Chapter 5 originated from Ilya Shapiro. Further, all aspects of the research, including implementation, were accomplished through joint efforts and collaboration.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from each of the co-authors to include the above materials in my dissertation. I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

II. Previous Publication

This dissertation includes two chapters that have been published as journal articles and one that is submitted to a journal for publication. See Table 0.0.1 below for more details:

Dissertation chapter	Article metadata	Publication status
Chapter 3	Curtis Bright, Amadou Keita, and Brett Stevens. Myrvold's results on orthogonal triples of 10×10 Latin squares: A SAT investigation. The Electronic Journal of Combinatorics, volume 33, issue 1, article P1.30, 2026, doi:10.37236/13960.	Published [34]
Chapter 4	Curtis Bright, Amadou Keita, and Brett Stevens. Orthogonal Latin squares of order 10 with two relations: A SAT investigation. Discrete Mathematics, Algorithms and Applications, published online November 2025. doi:10.1142/s1793830925501563.	In Press [33]
Chapter 5	Amadou Keita and Ilya Shapiro. Transitive sets of mutually orthogonal Latin squares. doi:10.48550/arXiv.2601.22205.	Submitted [69]

Table 0.0.1: Journal article publication and submission record.

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my dissertation. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I certify that, to the best of my knowledge, my dissertation does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my dissertation and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my dissertation, including any final revisions, as approved by my dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Mutually orthogonal Latin squares (MOLS) lie at the core of many classical combinatorial designs, and their existence and classification problems often sit at the boundary between structure and computational intractability. A central example is the long-standing open problem of the existence of a set of 3 MOLS of order 10. Motivated by this open problem, we develop computational techniques for constructing, classifying, and certifying the existence of a set of MOLS, and apply them in three complementary directions: SAT-based methods for structural constraints in a potential set of 3 MOLS of order 10, SAT-based certified exhaustive enumeration with linear-algebraic dependencies, and a SageMath-GAP integrated system for high symmetry/transitivity conditions on sets of MOLS using group-theoretic constraints to investigate MacNeish's conjecture.

The existence of an equivalence between a set of MOLS and a set of mutually transversal representation pairs is well-known, and it can be formulated via a composition operation on Latin squares. Myrvold considered a Latin square L, in a putative set of 3 MOLS of order 10, containing a 4×4 Latin subsquare and outlined 28 potential cases of existence. Myrvold determined that 20 of these cases do not exist. We encode an equivalent of orthogonality (transversal representation) in SAT that efficiently verifies the 20 nonexistence cases and resolves the remaining 8 open cases at the level of orthogonal pairs, with examples, whose extension to a triple would be L. Our results explain why the 8 remaining cases cannot be eliminated by analyzing orthogonal pairs and not involving L.

Howard outlined a characterization of a relation in a 4-net of order 10 (equivalently, a set of 2 MOLS of order 10), and Delisle used that outline to enumerate all 4-nets of order 10 with two nontrivial relations using custom-written code. We revisit the classification of a set of 2 MOLS of order 10 whose associated net possesses two nontrivial relations, i.e., $\mathbb{F}_2$-linear dependencies in its incidence matrix. We confirm prior exhaustive enumerations obtained by custom backtracking searches using an independent SAT-based enumeration. Beyond providing an alternate implementation,

the SAT approach yields independently checkable certificates of completeness, improving trust issues in the search code and enabling verification by external proof checkers. The resulting enumeration is also substantially more efficient, demonstrating the practical advantage of modern SAT solvers for large, highly constrained combinatorial searches.

MacNeish conjectured that for Latin squares of order n, the maximum number of Latin squares in a set of MOLS is one less than the smallest prime power divisor of n. This conjecture is false but we study it under symmetry restrictions, focusing on transitive and simply transitive sets of MOLS using group-theoretic constraints (via an integrated SageMath-GAP workflow) to analyze autotopy group and related symmetry actions and to support a targeted computer search. In the simply transitive setting, we show that the conjecture holds. For transitive sets of MOLS, we develop partial structural results. We also report that there are no transitive large counterexamples within the explored regimes. These results support the view that strong symmetry constraints substantially narrow the landscape of potential conjecture-violating sets of MOLS.

Together, the methods and findings in this dissertation sharpen the boundary of what is currently known about order 10, clarify which strategies have the potential to resolve the existence of a set of 3 MOLS of order 10, and contribute general tools for future work on transitivity, relations, construction, and certified enumeration in orthogonality problems.

DEDICATION

This dissertation is dedicated to my parents. They raised me from almost nothing—with hands that planted crops, reared farm animals, and worked through silence and hunger so we could survive. For most of my childhood, food was scarce. In those desperate times, even a thunderous rain would not stop my father from cutting wood in the forest to sell as firewood, or pedalling a bicycle through mud for miles to sell fish; the scorching sun would not stop my parents from working on the farm for hours; and deep hunger or the lack of resources would not shake their commitment to the education of me and my siblings. They were tried from many angles: even people close to us sought to deepen our hardship, make humiliation feel permanent, and repeatedly tried to obstruct my schooling. Yet my parents met what I could not bear with a patience greater than mine, and endured hardships deeper than any I have known—so that I might stand where I stand today.

I am blessed, not by my own doing; I am in debt, not owed; and I confess, with humility, that I cannot repay even the smallest imaginable fraction of the good my parents—and the rest of humanity—have shown me. May Allah, in Whose hand is my sustenance, reward them with honour, ease, and a reward that outlasts this world.

ACKNOWLEDGMENTS

I am deeply grateful to my advisors, Curtis Bright and Ilya Shapiro, as well as my de facto advisor, Brett Stevens, for their wisdom, high standards, insightful criticism, patience, and encouragement. Their guidance supported this work in innumerable ways, and I am thankful for their continued support—both academically and personally. Throughout my program, I benefited from their careful attention, prompt, and thoughtful feedback.

I am especially grateful to Curtis Bright for his steady mentorship. He took time to understand my strengths and weaknesses, provided research direction when I needed it, and read my work with exceptional care, improving it through attention to even the smallest details. His openness to collaboration and his consistent effort to track my progress helped me stay on course. I also gratefully acknowledge his additional research assistantship support, which provided both practical help and strong encouragement at a critical stage of my program. From him, I learned a great deal about mathematical writing, coding, and SAT-based computation, and I am sincerely thankful for everything.

I am grateful to Ilya Shapiro for pushing me to become more precise and disciplined in my mathematical communication. He encouraged me to present seminar talks so that I could learn to explain ideas with the level of clarity mathematics requires. His additional research support was a strong source of motivation. I also thank him for the many concepts I learned while working with him on Chapter 5 of this dissertation. I greatly admire his ability to move confidently between areas of study, and I hope to develop that breadth and fluency in my own work.

I thank Brett Stevens for his guidance and encouragement from the very beginning. His suggestions helped shape the direction that ultimately led to the topic of this dissertation, and his supportive, fatherly presence kept me encouraged through difficult moments. I am grateful to him for funding my trip to the 2023 Canadian Discrete and Algorithmic Mathematics Conference in Winnipeg, Canada.

I am grateful to Dennis Borisov for his genuine interest in my progress and well-

being. He repeatedly encouraged me to give seminar talks even if it may be outside my research area so that I could strengthen my mathematical precision through practice and feedback. His guidance and kindness helped make the department a welcoming and supportive place. I thank him sincerely for his mentorship and encouragement.

Sincere thanks and appreciation are due to Wendy Myrvold for insightful discussions at the 2023 Canadian Discrete and Algorithmic Mathematics Conference in Winnipeg, Canada. I am grateful to her for helping put many things into perspective and for challenging me to tackle the open problem on which this dissertation is based. I also acknowledge and thank Tanbir Ahmed for his assistance during the editing process of what became Chapter 3 of this dissertation.

I am indebted to the members of my dissertation committee for their constructive advice and insightful suggestions, which significantly enriched the content and quality of my dissertation. Special thanks go to the external examiner, Ian Wanless, for identifying gaps in the exposition, suggesting ways to address them, and providing valuable data for the classification results in Chapter 5. Ian Wanless's comments also clarified some known results, which helped streamline the necessary revisions.

I would also like to thank the former graduate secretaries and the current graduate secretary, Asma Naik, as well as the former administrative secretary, Lori Lano, for their support. They have been consistently kind, helpful, and efficient, and I am grateful for their guidance, insightful conversations, and timely responses whenever I needed assistance. The department professors have been kind, and I appreciate that.

I am grateful to Kathleen E. Lewis, one of my undergraduate professors, for the invaluable impact she has had on my life. More broadly, I acknowledge with gratitude the many teachers and school administrators who took a keen interest in my education and well-being from primary school onward. Their attention, encouragement, and care shaped my journey.

To every person who chose kindness when no one was watching: may Allah multiply your goodness, raise your ranks, and grant you the best of endings. Your financial support and encouragement made this dissertation possible, so I thank you from the bottom of my heart.

I am deeply grateful to my beloved parents and family for the patience, support, and encouragement they have given me from my early days through every stage of my education and academic formation. Their sacrifices made it possible for me to pursue my studies while they worked tirelessly on the farms. I owe a special debt of gratitude to my late cousin, Mankamang Badjie of Bansang, who took me into his home without expectation so that I could attend a good middle school; to my late aunt, Musukebba Jadama, who likewise hosted me with great generosity so that I could benefit from the opportunities of urban education; and to my visionary brother, Nfamara Keita, who lifted me from obscurity and provided me with opportunity, moral and financial support from grade seven to grade twelve, and extraordinary mentorship, enabling me to join the best class in the best high school in the country at that time. To all of you, I remain forever thankful for helping shape my future, preserve my freedom, and strengthen my sense of dignity. In truth, I have received so much from so little.

Finally, I acknowledge that all praise belongs to Allah—whose gifts reach us before we ask, who gives without measure, who replaces hardship with ease, and who brings help from places we never anticipated; whose mercy carries us when our strength is thin. To the Ever-Living Sustainer, the Bestower of goodness, the Originator of all that is seen and unseen, and the Lord of majesty and honor: I acknowledge Your countless blessings and offer this work with gratitude and trust, asking that it be accepted, made beneficial, and remembered with honor by later generations, and seeking refuge from pride, forgetfulness, and loss.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

$\mathbb{F}_2$	A field of order 2
k MOLS(n)	A set of k mutually orthogonal Latin squares of order n
k-net(n)	A k net of order n (equivalent to $k-2$ MOLS(n))
$N(n)$	The maximum size of a set of mutually orthogonal Latin squares of order n
CNF	Conjunctive normal form
CPU	Central processing unit (of a computer)
DRAT	Deletion resolution asymmetric tautology (SAT proof format / certificates)
GAP	Groups, Algorithms, and Programming (computer algebra system)
GF(q)	Galois field of order q
GiB	Gibibyte (of memory)
GHz	Gigahertz
MOLS	Mutually orthogonal Latin squares
OA	Orthogonal array
SAT	Boolean satisfiability
TRP	Transversal representation pair

CHAPTER 1

Introduction

Latin squares and their orthogonality lie at the interface of combinatorics, algebra, and computational search. A *Latin square* of order n is an $n \times n$ array with entries chosen from an n-set of *symbols*, such that each symbol appears exactly once in each row and exactly once in each column. Two Latin squares of order n are *orthogonal* if, when superimposed, every ordered pair of symbols occurs exactly once. A set of Latin squares is *mutually orthogonal* (MOLS) if every distinct pair in the set is orthogonal.

Existence and classification problems for sets of MOLS exhibit a characteristic mixture of structure and intractability. On the one hand, there are concrete constructions that explain why prime power orders admit large sets of MOLS. On the other hand, for composite orders, even basic existence questions can remain open for decades, and the search spaces for direct constructions can be astronomically large.

Write $N(n)$ for the maximum size of a set of MOLS of order n. Order 10 is a particularly important case. It is the smallest order for which the value of $N(n)$ is not known and is closely connected to the long history of existence/nonexistence questions for orthogonal arrays, nets, and finite geometries. For example, the nonexistence of a projective plane of order 10 was established by exhaustive computer search, ruling out a set of 9 MOLS of order 10. Since the prime power orders admit the maximal value $N(n) = n - 1$, this demonstrates that the order 10 behaves fundamentally differently from the prime power case. At the same time, many structured subclasses of pairs and triples of Latin squares of order 10 remain natural and meaningful targets for both theory and computation. One example is the existence of a Latin square with a subsquare of order 4 in a set of 3 MOLS of order 10.

This dissertation develops and applies a toolbox of computational constructions and classifications for sets of MOLS, guided by two recurring principles:

- **Exploitation of structure:** Effective searches require strong modelling choices and symmetry reduction. In our settings, structure appears in many forms: subsquares, transversal constraints, relations in nets, coset properties of groups, and transitivity properties of autotopy group actions.

- **Verifiable results:** Large scale searches become most valuable when they produce reusable artifacts: explicit examples, certified nonexistence proofs, and classification data invariant under natural equivalences like isotopy/main class.

A recurring theme in modern combinatorial computation is the translation of existence questions into constraint satisfaction problems with strong automated search engines. In this dissertation, the primary engines are SAT solving, SageMath, and GAP. In SAT, we encode Latin square constraints, orthogonality (or its equivalent formulations), and additional structure constraints as Boolean formulas in conjunctive normal form (CNF), and then leverage highly optimized solvers to search for solutions or prove unsatisfiability. We use GAP and SageMath to construct Latin squares and sets of MOLS from subgroup configurations.

Two aspects of the SAT approach are particularly important here:

- Latin squares and sets of MOLS come with large natural symmetry groups. Without symmetry breaking, a solver may waste most of its time rediscovering equivalent objects. The first two sets of research results in this dissertation include symmetry reductions tailored to the structure being studied.

- When a SAT solver reports that the constraints are unsatisfiable, the result, in Chapter 4, is accompanied by a proof trace (a DRAT certificate) that permits independent checking. For existence results, explicit solutions can be validated by separate code paths. These practices are essential whenever computational results are used as mathematical evidence.

1.1 Major results in this dissertation

Chapters 3 and 4 are research results on the problem of orthogonality for Latin squares of order 10, each with a concrete computational or structural deliverable. Chapter 5 is not restricted to order 10. This dissertation is organized around three sets of research results, each illustrating a different way in which structure can be used to make meaningful progress on problems of Latin squares and sets of MOLS, especially of order 10.

1.1.1 Order 10 via transversal representations

This research revisits a structured subproblem considered by Myrvold [89]: orthogonal triples of order 10 in which one of the Latin squares contains a 4×4 Latin subsquare. Myrvold derived necessary conditions in this setting and reduced the problem to a finite set of orthogonal pair types. A substantial portion of these types can be eliminated using constraints that involve only two squares at a time, leaving a small collection of computationally hard cases.

Our contribution is to explain why the two-square approach necessarily stalls: for each remaining pair type, we explicitly construct an orthogonal pair (or its equivalent formulation) satisfying the Myrvold framework via a transversal representation pair (TRP). Hence, determining the existence of a set of 3 MOLS of order 10 involving a Latin square with a subsquare of order 4 by our method, requires the constraints of all the three squares and their pairwise relationships. We also develop an efficient SAT encoding, based on a concise equivalence between orthogonality and transversal representations, that performs well in practice and produces explicit examples.

1.1.2 Enumeration of nets with two relations

A 4-net of order n is equivalent to an orthogonal pair of Latin squares of order n. Beyond orthogonality, nets can satisfy additional linear dependency constraints called relations, which come from dependencies in their incidence matrix over $\mathbb{F}_2$. For $n \equiv 2 \pmod 4$ (and in particular $n = 10$), the relations impose strong restrictions and are

tied to rank bounds of the matrix.

This research focuses on 4-nets of order 10 with at least two nontrivial relations. Howard [60] outlined them and Delisle [46] performed a computational enumeration of all such nets up to isomorphism. We recreate and independently verify the enumeration for the two-relation setting using SAT, and we strengthen the computational reliability by using checkable proof certificates where applicable. We also develop structural reasoning (via counting and linear-algebraic constraints on point types) that explains why the case of a certain parameter is impossible.

1.1.3 MacNeish's conjecture

This research is motivated by symmetry, not merely as a nuisance to be broken, but as a property to be classified. A set of q MOLS of order n corresponds to an orthogonal array with $q+2$ columns, and a natural notion of symmetry is the action of the autotopy group on the rows of this orthogonal array. We study transitive and simply transitive sets of MOLS via this action and formalize these notions within the orthogonal array framework.

We introduce group packets and prove a classification theorem: isotopy classes of sets of transitive q MOLS of order n correspond to equivalence classes of group packets. In the simply transitive setting, we obtain a sharp refinement identifying simply transitive sets of MOLS with disjoint group packets. We then apply this framework to MacNeish's conjecture under transitivity hypotheses, including reduction theorems driven by Sylow structure, and we prove that MacNeish's conjecture holds for simply transitive sets of MOLS. Finally, we implement a computational pipeline (SageMath with GAP) that searches for subgroup configurations in finite groups, constructs the induced Latin squares, tests associativity in a group-theoretic sense, and classifies non-associative outputs into main classes via canonical graph certificates.

1.2 Reproducibility

All three sets of research results are based on implementations that make the underlying mathematics explicit. On the SAT side, we use CNF encodings for Latin squares and orthogonality constraints together with symmetry breaking, and we validate results of unsatisfiable constraints via independent checking.

We use SageMath and GAP to enumerate and analyze Latin squares, and use canonical labelling (graph certificates) to reduce equivalence questions to tractable computations. A guiding goal throughout is that the output of a computation should be verifiable: either by producing explicit Latin squares, sets of MOLS, or subgroup data, or by producing checkable certificates supporting nonexistence claims.

1.3 Historical context of orthogonality

Combinatorial design theory is a fast-growing subject: each year brings hundreds of new results focused directly on designs, alongside a much larger body of work in statistics, algebra, and computer science that uses design-theoretic ideas as essential tools. The development of the field did not follow a single line; instead, it emerged from a mixture of recreational mathematics, concrete constructions, and later theoretical consolidation. Some very old objects, like magic squares, are known to be derived from Latin squares.

Definition 1.3.1. *A* magic square *of order n is an $n \times n$ array filled with the distinct integers, say, $0, 1, \ldots, n^2 - 1$, arranged so that the sum of the n entries in any row, any column, and each of the two main diagonals are the same constant called the* magic constant *[57, 72]*

$$M(n) = \frac{1}{n} \sum_{k=0}^{n^2-1} k = \frac{n(n^2-1)}{2}.$$

Let $L = \{l_{ij}\}$ be a Latin square of order n and $L' = \{l'_{ij}\}$ be another Latin square of order n both on the symbol set $S = \{0, 1, \ldots, n-1\}$. If L and L' are orthogonal

and both diagonal entries of L and L' are permutations of n elements, then a magic square M of size $n \times n$ can be constructed using the formula

$$M_{ij} = nl_{ij} + l'_{ij}.$$

An example of a magic square of order 4 (with the magic constant 30) following this construction is:

$$4\begin{bmatrix}0&3&1&2\\1&2&0&3\\2&1&3&0\\3&0&2&1\end{bmatrix} + \begin{bmatrix}0&3&2&1\\1&2&3&0\\3&0&1&2\\2&1&0&3\end{bmatrix} = \begin{bmatrix}0&15&6&9\\5&10&3&12\\11&4&13&2\\14&1&8&7\end{bmatrix}.$$

Early traces of Latin squares and their orthogonality appear in several cultural and mathematical settings. One of the earliest published sources is the monograph *Koo-Soo-Ryak* by Choi Seok-Jeong (1646–1715), where orthogonal Latin squares of order 9 are used to build a magic square, together with the observation that an orthogonal pair of order 10 was not found. Historical remarks [11] also point to medieval Islamic material (circa 1200) suggesting familiarity with Latin-square-like objects (amulets), including evidence consistent with two orthogonal 4×4 Latin squares. In eighteenth-century Europe, related ideas surfaced in recreational puzzles; for example, a card problem appearing in a later edition of Ozanam's [93] treatise is equivalent to constructing two orthogonal Latin squares of order 4.

The topic gained popularity through Euler's work on orthogonal Latin squares and magic squares from 1776 to 1782. In particular, Euler [50] highlighted the case of order 6 (the "36 officers" problem) and proposed the broader claim that orthogonal Latin squares should fail to exist whenever

$$n \equiv 2 \pmod 4.$$

Although the conjecture attracted attention for many decades, the order 6 case was only definitively resolved in 1900 by exhaustive enumeration by Tarry [106]. In 1934, Fisher and Yates [54] gave a shorter proof to the order 6 case. Several published

attempts, including Petersen [97] in 1901, Wernicke [117] in 1910, and MacNeish [79] in 1922, to settle Euler's general claim were later found to be erroneous. In 1942, Mann [80] considered the problem and gave a new construction technique of orthogonal Latin squares based on orthomorphisms, but did not address the conjecture.

Euler's conjecture was ultimately resolved in the mid-twentieth century: Bose and Shrikhande [25] produced a counterexample by constructing orthogonal Latin squares of order 22, and soon after, Bose, Shrikhande, and Parker [26] established that orthogonal Latin squares exist for every $n \equiv 2 \pmod 4$ except for $n = 2$ and $n = 6$.

The modern study of Latin squares grew out of concrete constructions and recreational problems, and it rapidly developed into a central theme of combinatorial design theory with deep links to statistics, algebra, and finite geometry. After the early appearances of orthogonality in the work of Choi Seok-Jeong and the subsequent prominence of Euler's questions, the subject entered a long period in which existence and nonexistence were settled order-by-order. The twentieth century brought a decisive conceptual shift: rather than treating orthogonal Latin squares as isolated puzzles, researchers began to organize them into systematic families and connect them with broader design objects such as orthogonal arrays and incidence geometries, especially through constructions over finite fields and groups [45, 52, 68, 104].

A key milestone in this transition is the construction of large sets of mutually orthogonal Latin squares. While the correspondence between complete sets of MOLS and finite projective planes—often referred to as the Bose equivalence theorem—is a cornerstone of modern finite geometry, the algebraic construction underlying these objects originates in the work of Moore [88]. In particular, for each prime power q, the finite field construction yields a set of

$$q - 1 \text{ MOLS of order } q,$$

which is maximal. This result forms the foundation for the classical equivalence between the existence of $n - 1$ MOLS of order n and finite projective (equivalently, affine) planes of order n [45]. This prime power theory shaped much of the postwar

development of design theory: it supplies abundant constructions, clarifies what an upper bound should be, and connects sets of MOLS to orthogonal arrays (and hence to experimental designs, coding theory, and related applications). Outside prime powers, however, the landscape is far less rigid: one sometimes patches together partial constructions (often via products, groups, or ad hoc combinatorial devices), and upper bounds are typically subtle.

The order $n = 10$ is a typical example of those that are not prime powers. It is large enough that brute-force enumeration is prohibitive without substantial structure, yet small enough to be the first genuinely stubborn case for several natural extremal questions about sets of MOLS. In particular, the existence of a set of 3 MOLS of order 10 remains one of the most visible open instances of the existence problem of sets of MOLS. This question is usually phrased as determining whether

$$N(10) \geq 3.$$

It is known that $N(10) \geq 2$ (orthogonal pairs exist) and that a set of 9 MOLS of order 10 does not exist [45, 75]. Coupling the nonexistence of a projective plane of order 10 [75] with a result of Bruck [38], we get

$$N(10) \leq 6,$$

which is currently the best upper bound known on $N(10)$.

In some recent investigations, one begins with objects (orthogonal pairs) that are known to exist and attempts to extend them by imposing the global consistency conditions encoded by orthogonality. Concretely, fixing a pair (P, Q) of orthogonal Latin squares of order 10, the task is to construct a third square L such that (P, L) and (Q, L) are orthogonal pairs. Constructing a third Latin square orthogonal to each of the two given squares is straightforward (if it exists); the bottleneck is the sheer number of candidate pairs one might have to examine. Even at order 10, this search space is typically overwhelming despite the strongest symmetry-breaking constraints currently available.

1.4 Structure of the dissertation

Chapter 2 develops background on the construction of sets of MOLS and the computational toolkit used throughout. Chapters 3, 4, and 5 present the three sets of research results described above: order 10 through transversal representations and Myrvold's framework, certified enumeration of nets with two nontrivial relations, and transitivity/group packets together with applications to MacNeish's conjecture. Chapter 6 concludes with a summary of the results and directions for future work.

Chapters 3–5 reproduce the papers developed during this Ph.D., and are included verbatim, with some formatting changes to ensure consistency, as part of this dissertation, with co-authorship and publication/submission details provided at the beginning of each chapter.

CHAPTER 2

Background

Mutually orthogonal Latin squares (MOLS) occupy a central position in combinatorial design theory and finite geometry. They simultaneously encode highly structured incidence systems—such as nets, affine and projective planes, and transversal designs—and provide a flexible language for translating algebraic ideas (groups, fields, and mappings) into concrete combinatorial objects. From a computational perspective, Latin squares are also a natural ground for modern constraint-solving methods: they have a compact definition, rich symmetry, and a search space that grows too quickly for naive enumeration, even at moderate orders n.

This dissertation is concerned with the construction and classification of sets of MOLS, with particular emphasis on the first truly difficult non-prime-power order, namely $n = 10$. At prime powers q, classical finite field construction yields a complete set of $q - 1$ MOLS of order q, which meets the universal upper bound $N(q) \leq q - 1$. Outside prime powers, the situation is markedly different: complete sets are not known to exist in general, and for $n = 10$ the existence of a set of three MOLS remains a prominent open problem. Progress, therefore, depends not only on finding new constructions but also on understanding how algebraic structure and symmetry interact with global orthogonality constraints.

The goal of this chapter is twofold. First, it establishes terminology and equivalence notions that recur throughout the dissertation: Latin squares and quasigroups, isotopy and autotopy, parastrophes and main classes, and the formal definition of orthogonality for pairs and larger sets. These notions provide the correct framework for comparing objects up to the transformations that preserve the underlying design. Second, it lays

out the computational viewpoint used in later chapters. We describe how Latin square and orthogonality constraints are encoded as Boolean satisfiability (SAT) instances, why different encodings behave differently in practice, and why symmetry-breaking is essential to make searches feasible at order 10.

2.1 Latin squares, orthogonality, and nets

Latin squares, and especially sets of mutually orthogonal Latin squares, provide a common source of many classical combinatorial designs, including nets, affine planes, projective planes, and transversal designs.

When an orthogonal set of Latin squares is derived from the Cayley table of a finite group by permuting columns, each square is uniquely determined by its first row (column, equivalently), which is a permutation of the group elements. This observation makes it possible to formulate and analyze orthogonality in purely algebraic terms, using notions such as difference matrices, complete mappings, and orthomorphisms. The associated designs obtained in this manner are then naturally characterized by the induced action of the underlying group on the resulting incidence structures.

2.1.1 Latin squares

Recall that:

Definition 2.1.1. *A* Latin square *of order n is an $n \times n$ matrix with entries chosen from an n-set of* symbols*, such that each symbol appears exactly once in each row and exactly once in each column.*

For Latin squares of order n, we will usually use the integers $0, 1, \ldots, n-1$ as indices for rows, columns, and symbols.

Definition 2.1.2. *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.*

Definition 2.1.3. *A* group *is an ordered pair* $(G, *)$*, where* G *is a set and* $*$ *is a binary operation on* G *satisfying the following axioms:*

1. ***Associativity:*** $(a * b) * c = a * (b * c)$ *for all* $a, b, c \in G$.
2. ***Identity:*** *There exists an element* $e \in G$*, called the* identity *of* G*, such that for all* $a \in G$ *we have* $a * e = e * a = a$.
3. ***Inverses:*** *For each* $a \in G$ *there exists an element* $a^{-1} \in G$*, called the* inverse *of* a*, such that* $a * a^{-1} = a^{-1} * a = e$.

Let $G = \{g_0, g_1, \ldots, g_{n-1}\}$ be a finite group. The Cayley table of G is the $n \times n$ matrix with (i, j)-entry equal to $g_i g_j$ for $0 \leq i, j \leq n - 1$. In 1854, Cayley [41, 42] noted that such a matrix is a Latin square; indeed, cancellation in groups implies each row and column is a permutation of G.

For some Latin squares that are equivalent to the Cayley table of a group, the group structure may not be immediately apparent.

One may need to relabel rows, columns, and symbols to obtain a more recognizable Cayley table. For example, the Latin square

$$L = \begin{bmatrix} 0 & 3 & 1 & 2 \\ 1 & 2 & 0 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 0 & 2 & 1 \end{bmatrix}$$

is isotopic to the Cayley table of the group $\mathbb{Z}_2 \times \mathbb{Z}_2$. Let

$$\sigma = (1\,3\,2), \qquad \tau = (1\,2\,3), \qquad \gamma = (1\,3\,2)$$

be permutations on the columns, rows, and symbols of L, respectively. We construct the Cayley table of the group $\mathbb{Z}_2 \times \mathbb{Z}_2$ considered as the Latin square L' with

$$L'_{\tau(i),\sigma(j)} = \gamma(L_{ij}) \qquad \text{for all } i, j,$$

i.e.,

$$L'_{r,c} = \gamma\big(L_{\tau^{-1}(r),\,\sigma^{-1}(c)}\big).$$

Applying first the column permutation σ replaces the columns in the order

$$[0, \sigma^{-1}(1), \sigma^{-1}(2), \sigma^{-1}(3)] = [0, 2, 3, 1],$$

so

$$\begin{bmatrix} 0 & 3 & 1 & 2 \\ 1 & 2 & 0 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 0 & 2 & 1 \end{bmatrix} \xrightarrow{\sigma} \begin{bmatrix} 0 & 2 & 3 & 1 \\ 1 & 3 & 2 & 0 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 0 & 2 \end{bmatrix}.$$

Next, applying the row permutation τ replaces the rows in the order

$$[0, \tau^{-1}(1), \tau^{-1}(2), \tau^{-1}(3)] = [0, 3, 1, 2],$$

giving

$$\begin{bmatrix} 0 & 2 & 3 & 1 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 0 & 2 \\ 1 & 3 & 2 & 0 \end{bmatrix}.$$

Finally, relabelling the symbols by $\gamma = (1\,3\,2)$, i.e.,

$$0 \mapsto 0, \quad 1 \mapsto 3, \quad 2 \mapsto 1, \quad 3 \mapsto 2,$$

yields

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} = L^{(\tau,\sigma,\gamma)} =: L',$$

which is the usual Cayley table of $\mathbb{Z}_2 \times \mathbb{Z}_2$ indexed by $0, 1, 2, 3$. We formalize this notion of equivalence in Section 2.1.1.1.1.

Definition 2.1.4 ([52, 68])**.** *A* quasigroup $(Q, \cdot)$ *is a set* $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ *with a binary operation* $\cdot$ *such that for each* $q_b, q_c \in Q$*, there exists a unique* $q_x \in Q$ *with* $q_x \cdot q_b = q_c$*, and for each* $q_a, q_c \in Q$*, there exists a unique* $q_y \in Q$ *with* $q_a \cdot q_y = q_c$*.*

Definition 2.1.5 ([52, 68])**.** *A quasigroup* $(Q, \cdot)$ *is a* loop *if it has an identity element* e *satisfying* $a \cdot e = e \cdot a = a$ *for all* $a \in Q$*.*

There is a useful correspondence between Latin squares and quasigroups. Let $Q = \{q_0, \ldots, q_{n-1}\}$ be a fixed labelled set, let $\mathcal{Q}$ be the set of quasigroups $(Q, \cdot)$ on Q, and let $\mathcal{L}$ be the set of Latin squares with rows, columns, and symbols indexed by Q. The map

$$f\colon \mathcal{Q} \to \mathcal{L}, \qquad f(Q, \cdot) = L = \{\, \ell_{q_r q_c} \,\}_{r,c=0}^{n-1}, \quad \ell_{q_r q_c} = q_r \cdot q_c, \tag{2.1.1}$$

is a bijection between $\mathcal{Q}$ and $\mathcal{L}$ [52, Thm. 1.1]. Without fixing such a labelling, this correspondence is only well-defined up to relabelling of Q, i.e., up to isomorphism. As a consequence, results about quasigroups translate directly to results about Latin squares and vice versa.

2.1.1.1 Equivalence

In the study of Latin squares, two squares are often regarded as "the same" if one can be obtained from the other by relabelling rows, columns, and symbols. This leads to the notion of isotopy. A second, coarser notion arises by allowing the three coordinates (row, column, symbol) to be permuted as well; this is called parastrophy. Combining these gives the main class (also called paratopy class or species).

Throughout this section, we illustrate the definitions using the following collection of order 7 Latin squares:

$$E_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 0 & 4 & 5 & 6 & 3 \\ 2 & 0 & 1 & 6 & 3 & 4 & 5 \\ 3 & 4 & 6 & 5 & 0 & 2 & 1 \\ 4 & 5 & 3 & 0 & 6 & 1 & 2 \\ 5 & 6 & 4 & 1 & 2 & 3 & 0 \\ 6 & 3 & 5 & 2 & 1 & 0 & 4 \end{bmatrix} \qquad E_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 0 & 6 & 3 & 4 & 5 \\ 2 & 0 & 1 & 4 & 5 & 6 & 3 \\ 3 & 5 & 6 & 1 & 0 & 2 & 4 \\ 4 & 6 & 5 & 0 & 2 & 3 & 1 \\ 5 & 4 & 3 & 2 & 6 & 1 & 0 \\ 6 & 3 & 4 & 5 & 1 & 0 & 2 \end{bmatrix}$$

$$E_3 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 0 & 6 & 3 & 4 & 5 \\ 2 & 0 & 6 & 1 & 5 & 3 & 4 \\ 3 & 4 & 1 & 5 & 2 & 6 & 0 \\ 4 & 5 & 3 & 0 & 6 & 2 & 1 \\ 5 & 6 & 4 & 2 & 0 & 1 & 3 \\ 6 & 3 & 5 & 4 & 1 & 0 & 2 \end{bmatrix} \qquad E_4 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 0 & 4 & 5 & 6 & 3 \\ 2 & 0 & 1 & 6 & 3 & 4 & 5 \\ 3 & 4 & 6 & 5 & 0 & 1 & 2 \\ 4 & 5 & 3 & 0 & 6 & 2 & 1 \\ 5 & 6 & 4 & 2 & 1 & 3 & 0 \\ 6 & 3 & 5 & 1 & 2 & 0 & 4 \end{bmatrix}$$

$$E_5 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 0 & 1 & 6 & 3 & 4 & 5 \\ 1 & 2 & 0 & 4 & 5 & 6 & 3 \\ 4 & 6 & 5 & 0 & 1 & 3 & 2 \\ 3 & 5 & 6 & 2 & 0 & 1 & 4 \\ 6 & 3 & 4 & 5 & 2 & 0 & 1 \\ 5 & 4 & 3 & 1 & 6 & 2 & 0 \end{bmatrix} \qquad E_6 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 0 & 1 & 6 & 3 & 4 & 5 \\ 1 & 2 & 0 & 4 & 5 & 6 & 3 \\ 4 & 5 & 6 & 0 & 1 & 3 & 2 \\ 3 & 6 & 5 & 2 & 0 & 1 & 4 \\ 6 & 4 & 3 & 5 & 2 & 0 & 1 \\ 5 & 3 & 4 & 1 & 6 & 2 & 0 \end{bmatrix}$$

$$E_7 = \begin{bmatrix} 0 & 2 & 1 & 4 & 3 & 6 & 5 \\ 1 & 0 & 2 & 6 & 5 & 3 & 4 \\ 2 & 1 & 0 & 5 & 6 & 4 & 3 \\ 3 & 6 & 4 & 0 & 2 & 5 & 1 \\ 4 & 3 & 5 & 1 & 0 & 2 & 6 \\ 5 & 4 & 6 & 3 & 1 & 0 & 2 \\ 6 & 5 & 3 & 2 & 4 & 1 & 0 \end{bmatrix} \qquad E_8 = \begin{bmatrix} 0 & 2 & 1 & 4 & 3 & 6 & 5 \\ 1 & 0 & 2 & 5 & 6 & 4 & 3 \\ 2 & 1 & 0 & 6 & 5 & 3 & 4 \\ 3 & 6 & 4 & 0 & 2 & 5 & 1 \\ 4 & 3 & 5 & 1 & 0 & 2 & 6 \\ 5 & 4 & 6 & 3 & 1 & 0 & 2 \\ 6 & 5 & 3 & 2 & 4 & 1 & 0 \end{bmatrix}$$

2.1.1.1.1 Isotopy Let $L = \{\ell_{ij}\}$ and $L' = \{\ell'_{ij}\}$ be Latin squares of order n on the same symbol set S.

Definition 2.1.6 ([52, 68])**.** *We say that L' is* isotopic *to L if there exist bijections*

$$\alpha, \beta \colon \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}, \qquad \gamma \colon S \to S$$

such that

$$\ell'_{\alpha(i),\,\beta(j)} = \gamma(\ell_{ij}) \qquad \textit{for all } i, j.$$

The triple (α, β, γ) is an isotopy *from L to L'. If $L' = L$, then (α, β, γ) is an* autotopy *of L.*

Isotopy is an equivalence relation on the set of Latin squares of order n. Each isotopy class can be large: there are $n!$ possible row permutations, $n!$ column permutations,

and $n!$ symbol permutations, so an isotopy class has size at most $(n!)^3$ (often smaller due to symmetries).

Our set of Latin squares belongs to exactly three isotopy classes:

$$\{E_1, E_4, E_5, E_6, E_7, E_8\} \quad \text{and} \quad \{E_2\} \quad \text{and} \quad \{E_3\}.$$

The notion of isotopy translates directly to quasigroups under the bijection f in (2.1.1).

Definition 2.1.7 ([52, 68])**.** *Two quasigroups $(Q, \cdot)$ and $(Q', \circ)$ are* isotopic *if there exist bijections $\alpha, \beta, \gamma \colon Q \to Q'$ such that*

$$\gamma(a \cdot b) = \alpha(a) \circ \beta(b) \qquad \text{for all } a, b \in Q.$$

An autotopy *is an isotopy from $(Q, \cdot)$ to itself. When $\alpha = \beta = \gamma$, the condition becomes $\alpha(a \cdot b) = \alpha(a) \cdot \alpha(b)$; in this case α is an* automorphism*.*

A classical fact is that every quasigroup is isotopic to a loop, equivalently, every Latin square is isotopic to a "reduced" one [52].

Definition 2.1.8 ([52, 68])**.** *A Latin square L on symbols $S = \{0, 1, \ldots, n-1\}$ is* reduced *(or in* standard form*) if its first row and first column are in ascending order:*

$$L(0, j) = j \quad \text{and} \quad L(i, 0) = i \qquad (0 \le i, j \le n-1).$$

Definition 2.1.9 ([52])**.** *Let G be a finite group. A Latin square is* based on the group *G if it is isotopic to the Cayley table of G.*

2.1.1.1.2 Parastrophy It is often convenient to represent a Latin square $L = \{\ell_{ij}\}$ by its set of *triples*

$$T_L = \{\, (i, j, \ell_{ij}) : 0 \le i, j \le n-1\} \subseteq \{0, 1, \ldots, n-1\,\}^3,$$

whose coordinates are (row, column, symbol). An isotopy (α, β, γ) acts coordinatewise on triples by

$$(i, j, \ell_{ij}) \longmapsto \big(\alpha(i), \beta(j), \gamma(\ell_{ij})\big),$$

so L and L' are isotopic if and only if

$$T_{L'} = \{\, (\alpha(i), \beta(j), \gamma(\ell_{ij})) : (i, j, \ell_{ij}) \in T_L \,\}.$$

Parastrophy enlarges the equivalence relation by allowing permutations of the three coordinates themselves.

Definition 2.1.10 ([68]). *For a permutation $\pi \in S_3$, the* π-parastrophe L^π *is defined by*

$$T_{L^\pi} = \{\, \pi(r, c, s) : (r, c, s) \in T_L \,\}.$$

Equivalently, if $\pi(r, c, s) = (r', c', s')$, then $L^\pi(r', c') = s'$.

Since $|S_3| = 6$, each Latin square has six parastrophes. Writing a permutation in one-line form as (a, b, c), meaning $(r, c, s) \mapsto (x_a, x_b, x_c)$ with $(x_0, x_1, x_2) = (r, c, s)$, we obtain:

1. $(0, 1, 2)$: the identity parastrophe, so $L^{(0,1,2)} = L$.
2. $(1, 0, 2)$: swap rows and columns, so $L^{(1,0,2)} = L^{\mathsf{T}}$ (matrix transpose), since

$$L(r, c) = s \iff L^{(1,0,2)}(c, r) = s.$$

3. $(0, 2, 1)$: swap columns and symbols, so

$$L(r, c) = s \iff L^{(0,2,1)}(r, s) = c.$$

   Given r and s, it returns the unique c with $L(r, c) = s$.

4. $(2, 1, 0)$: swap rows and symbols, so

$$L(r, c) = s \iff L^{(2,1,0)}(s, c) = r.$$

Given s and c, it returns the unique r with $L(r, c) = s$.

5. $(1, 2, 0)$: send $(r, c, s) \mapsto (c, s, r)$, so

$$L(r, c) = s \iff L^{(1,2,0)}(c, s) = r.$$

6. $(2, 0, 1)$: send $(r, c, s) \mapsto (s, r, c)$, so

$$L(r, c) = s \iff L^{(2,0,1)}(s, r) = c.$$

For computational purposes, it is helpful to observe that the $(0, 2, 1)$-parastrophe is obtained by inverting each row permutation of L; we call it the *row inverse* of L. Similarly, the $(2, 1, 0)$-parastrophe is obtained by inverting each column permutation of L; we call it the *column inverse*. These can be used as intermediaries to compute the remaining parastrophes:

$$(0, 1, 2) \xrightarrow{\text{row inverse}} (0, 2, 1) \xrightarrow{\text{transpose}} (2, 0, 1)$$

$$(0, 1, 2) \xrightarrow{\text{column inverse}} (2, 1, 0) \xrightarrow{\text{transpose}} (1, 2, 0).$$

Example 2.1.11. *An example of a Latin square E_1 and its six parastrophes is:*

$$(0, 1, 2)\text{-}parastrophe : E_1$$
$$(1, 0, 2)\text{-}parastrophe : E_4$$
$$(0, 2, 1)\text{-}parastrophe : E_5$$
$$(1, 2, 0)\text{-}parastrophe : E_6$$
$$(2, 0, 1)\text{-}parastrophe : E_7$$
$$(2, 1, 0)\text{-}parastrophe : E_8$$

2.1.1.1.3 Main class

Definition 2.1.12 ([68])**.** *The* main class *of a Latin square L is the set of all*

parastrophes of all isotopes of L [68, Ch. I].

Our set of Latin squares belongs to exactly two main classes:

$$\{E_1, E_2, E_4, E_5, E_6, E_7, E_8\} \quad \text{and} \quad \{E_3\}.$$

For an orthogonal pair of Latin squares (L, L'), one can similarly represent them by a set of quadruples, and there are $4! = 24$ coordinate permutations (parastrophes). In practice, the main class equivalence for orthogonal pairs is tested via canonical graph representations (see, e.g., [49, 52, 68]).

2.1.2 Orthogonality

Definition 2.1.13 ([52])**.** *The Latin squares L and L' of order n on a symbol set S are* orthogonal *if for each $a \in S$ and $b \in S$ there exists a unique pair (i, j) such that $\ell_{ij} = a$ and $\ell'_{ij} = b$. In this case, L' is an* orthogonal mate *of L.*

Example 2.1.14. *An example of an orthogonal pair of Latin squares of order* 10*.*

$$L = \begin{bmatrix} 3 & 5 & 1 & 7 & 2 & 0 & 6 & 4 & 8 & 9 \\ 6 & 0 & 2 & 9 & 1 & 3 & 4 & 7 & 5 & 8 \\ 0 & 1 & 4 & 3 & 8 & 2 & 5 & 6 & 9 & 7 \\ 2 & 9 & 3 & 1 & 0 & 8 & 7 & 5 & 4 & 6 \\ 1 & 3 & 0 & 2 & 4 & 7 & 8 & 9 & 6 & 5 \\ 5 & 2 & 6 & 0 & 3 & 1 & 9 & 8 & 7 & 4 \\ 7 & 4 & 8 & 5 & 9 & 6 & 0 & 1 & 2 & 3 \\ 8 & 6 & 9 & 4 & 7 & 5 & 1 & 2 & 3 & 0 \\ 4 & 7 & 5 & 8 & 6 & 9 & 2 & 3 & 0 & 1 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 0 & 1 & 2 \end{bmatrix} \qquad L' = \begin{bmatrix} 0 & 7 & 8 & 1 & 6 & 4 & 2 & 5 & 3 & 9 \\ 7 & 1 & 9 & 0 & 5 & 2 & 3 & 4 & 6 & 8 \\ 3 & 4 & 2 & 6 & 0 & 5 & 9 & 1 & 8 & 7 \\ 4 & 2 & 5 & 3 & 9 & 1 & 8 & 0 & 7 & 6 \\ 2 & 8 & 6 & 7 & 1 & 0 & 4 & 3 & 9 & 5 \\ 1 & 3 & 0 & 8 & 7 & 9 & 6 & 2 & 5 & 4 \\ 9 & 0 & 7 & 2 & 4 & 8 & 5 & 6 & 1 & 3 \\ 6 & 5 & 1 & 9 & 2 & 3 & 7 & 8 & 4 & 0 \\ 8 & 6 & 4 & 5 & 3 & 7 & 0 & 9 & 2 & 1 \\ 5 & 9 & 3 & 4 & 8 & 6 & 1 & 7 & 0 & 2 \end{bmatrix}$$

Orthogonality is preserved by applying the same row permutation to both squares and likewise the same column permutation to both squares. Symbol permutations may be applied independently to each square without destroying orthogonality. In contrast, permuting rows (or columns) in one single square can destroy orthogonality.

Example 2.1.15. *Consider the orthogonal mates L and L' in Example 2.1.14.*

Independent row permutation. *Permuting the first two rows of L only (and leaving L' unchanged) gives*

$$\widetilde{L} = \begin{bmatrix} 6 & 0 & 2 & 9 & 1 & 3 & 4 & 7 & 5 & 8 \\ 3 & 5 & 1 & 7 & 2 & 0 & 6 & 4 & 8 & 9 \\ 0 & 1 & 4 & 3 & 8 & 2 & 5 & 6 & 9 & 7 \\ 2 & 9 & 3 & 1 & 0 & 8 & 7 & 5 & 4 & 6 \\ 1 & 3 & 0 & 2 & 4 & 7 & 8 & 9 & 6 & 5 \\ 5 & 2 & 6 & 0 & 3 & 1 & 9 & 8 & 7 & 4 \\ 7 & 4 & 8 & 5 & 9 & 6 & 0 & 1 & 2 & 3 \\ 8 & 6 & 9 & 4 & 7 & 5 & 1 & 2 & 3 & 0 \\ 4 & 7 & 5 & 8 & 6 & 9 & 2 & 3 & 0 & 1 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 0 & 1 & 2 \end{bmatrix} \qquad L' = \begin{bmatrix} 0 & 7 & 8 & 1 & 6 & 4 & 2 & 5 & 3 & 9 \\ 7 & 1 & 9 & 0 & 5 & 2 & 3 & 4 & 6 & 8 \\ 3 & 4 & 2 & 6 & 0 & 5 & 9 & 1 & 8 & 7 \\ 4 & 2 & 5 & 3 & 9 & 1 & 8 & 0 & 7 & 6 \\ 2 & 8 & 6 & 7 & 1 & 0 & 4 & 3 & 9 & 5 \\ 1 & 3 & 0 & 8 & 7 & 9 & 6 & 2 & 5 & 4 \\ 9 & 0 & 7 & 2 & 4 & 8 & 5 & 6 & 1 & 3 \\ 6 & 5 & 1 & 9 & 2 & 3 & 7 & 8 & 4 & 0 \\ 8 & 6 & 4 & 5 & 3 & 7 & 0 & 9 & 2 & 1 \\ 5 & 9 & 3 & 4 & 8 & 6 & 1 & 7 & 0 & 2 \end{bmatrix}.$$

This pair is not orthogonal: the ordered pair $(6, 0)$ appears twice, at cells $(0, 0)$ and $(5, 2)$ (highlighted).

Independent column permutation. *Permuting the first two columns of L only (and leaving L' unchanged) gives*

$$\widehat{L} = \begin{bmatrix} 5 & 3 & 1 & 7 & 2 & 0 & 6 & 4 & 8 & 9 \\ 0 & 6 & 2 & 9 & 1 & 3 & 4 & 7 & 5 & 8 \\ 1 & 0 & 4 & 3 & 8 & 2 & 5 & 6 & 9 & 7 \\ 9 & 2 & 3 & 1 & 0 & 8 & 7 & 5 & 4 & 6 \\ 3 & 1 & 0 & 2 & 4 & 7 & 8 & 9 & 6 & 5 \\ 2 & 5 & 6 & 0 & 3 & 1 & 9 & 8 & 7 & 4 \\ 4 & 7 & 8 & 5 & 9 & 6 & 0 & 1 & 2 & 3 \\ 6 & 8 & 9 & 4 & 7 & 5 & 1 & 2 & 3 & 0 \\ 7 & 4 & 5 & 8 & 6 & 9 & 2 & 3 & 0 & 1 \\ 8 & 9 & 7 & 6 & 5 & 4 & 3 & 0 & 1 & 2 \end{bmatrix} \qquad L' = \begin{bmatrix} 0 & 7 & 8 & 1 & 6 & 4 & 2 & 5 & 3 & 9 \\ 7 & 1 & 9 & 0 & 5 & 2 & 3 & 4 & 6 & 8 \\ 3 & 4 & 2 & 6 & 0 & 5 & 9 & 1 & 8 & 7 \\ 4 & 2 & 5 & 3 & 9 & 1 & 8 & 0 & 7 & 6 \\ 2 & 8 & 6 & 7 & 1 & 0 & 4 & 3 & 9 & 5 \\ 1 & 3 & 0 & 8 & 7 & 9 & 6 & 2 & 5 & 4 \\ 9 & 0 & 7 & 2 & 4 & 8 & 5 & 6 & 1 & 3 \\ 6 & 5 & 1 & 9 & 2 & 3 & 7 & 8 & 4 & 0 \\ 8 & 6 & 4 & 5 & 3 & 7 & 0 & 9 & 2 & 1 \\ 5 & 9 & 3 & 4 & 8 & 6 & 1 & 7 & 0 & 2 \end{bmatrix}.$$

Again, the pair is not orthogonal: the ordered pair $(5, 0)$ appears twice, at cells $(0, 0)$ and $(3, 7)$ (highlighted).

The following theorem formalizes the fact that applying the *same* row and column permutations to both squares preserves orthogonality (while allowing independent symbol permutations).

Theorem 2.1.16 ([52])**.** *Let L_1 and L_2 be Latin squares of the same order on a symbol set S. Suppose $(\alpha, \beta, \gamma_1)$ is an isotopy from L_1 to L_1' and $(\alpha, \beta, \gamma_2)$ is an isotopy from*

L_2 to L_2'. Then L_1 is orthogonal to L_2 if and only if L_1' is orthogonal to L_2'.

Proof. Write $L_k = \{\ell_{kij}\}$ and $L_k' = \{\ell'_{kij}\}$ for $k \in \{1, 2\}$. By the definition of isotopy, for all i, j we have

$$\ell'_{1\alpha(i)\,\beta(j)} = \gamma_1(\ell_{1ij}) \qquad \text{and} \qquad \ell'_{2\alpha(i)\,\beta(j)} = \gamma_2(\ell_{2ij}).$$

Assume L_1 is orthogonal to L_2. Fix symbols $a' \in S$ and $b' \in S$. Let $a = \gamma_1^{-1}(a')$ and $b = \gamma_2^{-1}(b')$. Orthogonality of L_1 and L_2 gives a unique pair (i, j) such that $\ell_{1ij} = a$ and $\ell_{2ij} = b$. Set $i' = \alpha(i)$ and $j' = \beta(j)$. Then

$$\ell'_{1i'j'} = \gamma_1(\ell_{1ij}) = \gamma_1(a) = a' \quad \text{and} \quad \ell'_{2i'j'} = \gamma_2(\ell_{2ij}) = \gamma_2(b) = b'.$$

Uniqueness follows because α, β are bijections. The converse follows by applying the same argument to the inverse isotopies. □

Definition 2.1.17 ([52, 68])**.** *A set of Latin squares in which each pair is orthogonal is a set of* mutually orthogonal set of Latin squares *(a set of* MOLS*).*

A set of MOLS is *maximal* if it cannot be extended by adding another Latin square orthogonal to all members. Let $N(n)$ denote the maximum size of a set of MOLS of order n. It is known that $N(n) \leq n - 1$ for all $n \geq 2$.

To see this, let $L_0, L_1, \ldots, L_{r-1}$ be a set of r MOLS of order n on $S = \{0, 1, \ldots, n-1\}$. For each j, we may apply a symbol permutation to obtain Latin squares L_j' such that

$$L_j'[i, 0] = i \qquad \text{for all } i = 0, 1, \ldots, n-1 \text{ and for all } j.$$

Now consider the entries $L_j'[0, 1]$. Since $L_j'[0, 0] = 0$, we have $L_j'[0, 1] \neq 0$ for each j. Suppose $L_j'[0, 1] = L_k'[0, 1]$ for some $j \neq k$ and let $i = L_j'[0, 1]$. Then $(L_j'[0, 1], L_k'[0, 1]) = (i, i)$ and $(L_j'[i, 0], L_k'[i, 0]) = (i, i)$, so we have equality of the pairs

$$(L_j'[i, 0], L_k'[i, 0]) = (L_j'[0, 1], L_k'[0, 1]),$$

which contradicts our assumption of pairwise orthogonality. Thus, the set of values

$L'_0[0,1], \ldots, L'_{r-1}[0,1]$ are pairwise distinct. Hence,

$$\{\, L'_j[0,1] : 0 \le j \le r-1 \,\} \subseteq \{1, 2, \ldots, n-1\}$$

has size r, implying $r \le n-1$.

The upper bound is attained at prime powers.

Theorem 2.1.18 (Finite field construction)**.** *Let*

$$\mathrm{GF}(q) = \{x_0, x_1, \ldots, x_{q-1}\}$$

be the finite field of order q. For each $b \in \mathrm{GF}(q)$, define the $q \times q$ matrix L_b by

$$L_{bij} = x_i + bx_j \qquad (0 \le i, j \le q-1).$$

Then for each $b \in \mathrm{GF}(q) \setminus \{0\}$, the matrix L_b is a Latin square of order q, and the family

$$\{\, L_b : b \in \mathrm{GF}(q) \setminus \{0\} \,\}$$

is a set of $q-1$ MOLS of order q [81].

Example 2.1.19. *For $q = 7$, identify $\mathrm{GF}(7) = \{0,1,2,3,4,5,6\}$ and define*

$$L_{bij} = i + bj \pmod 7 \qquad \text{for all } b \in \{1,2,3,4,5,6\}.$$

Then $\{L_1, L_2, L_3, L_4, L_5, L_6\}$ is a set of 6 MOLS of order 7. The concrete representatives are:

$$L_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 0 \\ 2 & 3 & 4 & 5 & 6 & 0 & 1 \\ 3 & 4 & 5 & 6 & 0 & 1 & 2 \\ 4 & 5 & 6 & 0 & 1 & 2 & 3 \\ 5 & 6 & 0 & 1 & 2 & 3 & 4 \\ 6 & 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \qquad L_2 = \begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 \\ 1 & 3 & 5 & 0 & 2 & 4 & 6 \\ 2 & 4 & 6 & 1 & 3 & 5 & 0 \\ 3 & 5 & 0 & 2 & 4 & 6 & 1 \\ 4 & 6 & 1 & 3 & 5 & 0 & 2 \\ 5 & 0 & 2 & 4 & 6 & 1 & 3 \\ 6 & 1 & 3 & 5 & 0 & 2 & 4 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} 0 & 3 & 6 & 2 & 5 & 1 & 4 \\ 1 & 4 & 0 & 3 & 6 & 2 & 5 \\ 2 & 5 & 1 & 4 & 0 & 3 & 6 \\ 3 & 6 & 2 & 5 & 1 & 4 & 0 \\ 4 & 0 & 3 & 6 & 2 & 5 & 1 \\ 5 & 1 & 4 & 0 & 3 & 6 & 2 \\ 6 & 2 & 5 & 1 & 4 & 0 & 3 \end{bmatrix} \qquad L_4 = \begin{bmatrix} 0 & 4 & 1 & 5 & 2 & 6 & 3 \\ 1 & 5 & 2 & 6 & 3 & 0 & 4 \\ 2 & 6 & 3 & 0 & 4 & 1 & 5 \\ 3 & 0 & 4 & 1 & 5 & 2 & 6 \\ 4 & 1 & 5 & 2 & 6 & 3 & 0 \\ 5 & 2 & 6 & 3 & 0 & 4 & 1 \\ 6 & 3 & 0 & 4 & 1 & 5 & 2 \end{bmatrix}$$

$$L_5 = \begin{bmatrix} 0 & 5 & 3 & 1 & 6 & 4 & 2 \\ 1 & 6 & 4 & 2 & 0 & 5 & 3 \\ 2 & 0 & 5 & 3 & 1 & 6 & 4 \\ 3 & 1 & 6 & 4 & 2 & 0 & 5 \\ 4 & 2 & 0 & 5 & 3 & 1 & 6 \\ 5 & 3 & 1 & 6 & 4 & 2 & 0 \\ 6 & 4 & 2 & 0 & 5 & 3 & 1 \end{bmatrix} \qquad L_6 = \begin{bmatrix} 0 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}.$$

For non-prime power orders n, it is not known in general whether $N(n)$ can equal $n-1$. The first difficult order from this standpoint is $n = 10$. Orthogonal pairs of order 10 exist (so $N(10) \geq 2$), but the existence of a set of three MOLS of order 10 remains open.

Upper bounds on $N(10)$ are tied to bounds on the size of a net and to the nonexistence of a projective plane of order 10. In particular, the nonexistence of a projective plane of order 10 was established by Lam et al. [75]. Combining this with a theorem of Bruck [38] relating nets and projective planes yields the best currently known general bound

$$N(10) \leq 6.$$

2.1.3 Nets

Definition 2.1.20. *A k-net of order n, denoted k-net(n), is a finite incidence structure with n^2 points and kn lines such that:*

1. *each line is incident with exactly n points;*
2. *each point is incident with exactly k lines;*
3. *the lines are partitioned into k parallel classes, each consisting of n pairwise disjoint lines; and*

4. any two lines from distinct parallel classes meet in exactly one point.

Definition 2.1.21. *An* incidence matrix *of a k-net(n) is the $n^2 \times kn$ matrix over* GF(2) *whose (i, j) entry is* 1 *if the ith point lies on the jth line and* 0 *otherwise.*

Lines can be ordered by parallel class so that the first n columns correspond to the first class, the next n to the second, and so on. There is a well-known equivalence between k-nets of order n and sets of $(k-2)$ mutually orthogonal Latin squares of order n [40, 68]. Under this correspondence (and up to main class equivalence), we may assume the first parallel class encodes rows, the second encodes columns, and, for $\ell > 2$, the ℓth parallel class encodes the symbols of the $(\ell-2)$th Latin square in a set of $(k-2)$ MOLS of order n. In a 3-net of order n, a triple (i, j, s) corresponds to a point incident with the ith row line, the jth column line, and the sth symbol line. A point in a k-net of order n is a coordinatization of the corresponding set of $k-2$ MOLS of order n as $(i, j, s_1, \ldots, s_{k-2})$ where i is the row line, j is the column line, and for $1 \leq t \leq k-2$, s_t is the symbol line of the Latin square L_t in the corresponding set of $k-2$ MOLS of order n.

Definition 2.1.22. *Let (A, B) be an orthogonal pair of order n. Let $K = [R \mid C \mid S \mid S']$ be the associated $n^2 \times 4n$ incidence matrix, where R encodes row lines, C column lines, S the symbol lines of A, and S' the symbol lines of B. The* binary code *of the corresponding* 4*-net is the column space of K over* GF(2)*. The* dimension *of this code is the rank of K over* GF(2)*.*

The following definition is central in our investigation in Chapter 4.

Definition 2.1.23. *A* relation *in the binary code of an orthogonal pair of Latin squares of order n is a subset of columns whose sum over* GF(2) *is the zero vector.*

Relations formed by the sum of an even number of entire parallel classes are called *trivial*. For a 4-net, the number of trivial relations equals

$$\binom{4}{0} + \binom{4}{2} + \binom{4}{4} = 8,$$

corresponding to choosing 0, 2, or 4 whole parallel classes.

Definition 2.1.24. *Given a relation X and a subset $J \subseteq \{R, C, S, S'\}$ of even cardinality, we define the* complement *of X with respect to J to be the symmetric difference*

$$X^{(J)} := X \triangle \bigcup_{N \in J} N,$$

that is, inside each parallel class $N \in J$ we replace the chosen lines of X by their set-theoretic complement in N.

Every point of the net lies on exactly one line from each parallel class, so toggling membership in an even number of classes changes the incidence multiplicity at each point by an even integer. Hence $X^{(J)}$ is again a relation. We regard relations that differ by such a complementation as equivalent.

Howard [60, Prop. 5.4] showed that if $n \equiv 2 \pmod 4$ and a relation contains at most $n/2$ columns from each of R, C, and S, then it contains exactly k columns from each of R, C, S, and S', where k is even and

$$\frac{n}{3} \leq k < \frac{n}{2}.$$

For $n = 10$ this forces $k = 4$. It follows that every nontrivial relation is equivalent, under complementation by whole parallel classes, to one containing exactly 4 lines in each parallel class.

2.2 Boolean logic

In this section, we provide basic preliminaries on Boolean logic and satisfiability (SAT) solving of sets of MOLS. We will build on this foundation to construct orthogonal Latin squares in Chapters 3 and 4.

2.2.1 Basics of Boolean logic

Definition 2.2.1. *A proposition is a declarative statement that has a definite truth value: it is either true or false.*

Instead of just a proposition, we will sometimes use a *(Boolean) propositional variable*, or just a *Boolean variable*, or *variable* if there is no confusion.

Definition 2.2.2. *A* negation *of a Boolean variable x, denoted $\neg x$, is the change of its truth value to its opposite.*

Definition 2.2.3. *A* literal *is either a Boolean variable x or its negation $\neg x$.*

Definition 2.2.4. *A* disjunction *of two Boolean variables x and y, denoted $x \vee y$, is the variable defined by its truth function:*

$$x \vee y \ := \begin{cases} \textit{true} & \textit{if } x \textit{ is true or } y \textit{ is true,} \\ \textit{false} & \textit{if } x \textit{ is false and } y \textit{ is false.} \end{cases}$$

Definition 2.2.5. *A* conjunction *of two Boolean variables x and y, denoted $x \wedge y$, is the variable defined by its truth function:*

$$x \wedge y \ := \begin{cases} \textit{true} & \textit{if } x \textit{ is true and } y \textit{ is true,} \\ \textit{false} & \textit{otherwise.} \end{cases}$$

The Boolean formula $x \vee y$ is logically equivalent to $\neg(\neg x \wedge \neg y)$ and the Boolean formula $x \wedge y$ is logically equivalent to $\neg(\neg x \vee \neg y)$.

Definition 2.2.6. *A* clause *is a disjunction of literals, i.e.,*

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k,$$

where each ℓ_i is a literal.

Definition 2.2.7. *A* SAT solver *is a program that determines whether there exists a truth assignment to the variables of a Boolean formula under which the formula evaluates to* true.

In practice, SAT solvers expect input formulas in *conjunctive normal form* (CNF). A CNF formula is built from Boolean variables using the connectives $\wedge$ (and), $\vee$ (or), and $\neg$ (not).

Definition 2.2.8. *A* CNF formula *is a conjunction of clauses, i.e.,*

$$c_1 \wedge c_2 \wedge \cdots \wedge c_m,$$

where each c_j is a clause.

A CNF formula is satisfied exactly when every clause is satisfied and a clause is satisfied exactly when at least one of its literals is true. (An empty conjunction is true; an empty disjunction is false.)

It is often convenient to treat a clause as a set of literals (since disjunction is commutative), writing

$$\ell_1 \vee \ell_2 \vee \ell_3 \quad \text{as} \quad \{\ell_1, \ell_2, \ell_3\}.$$

Under this view, a CNF formula may be identified with a set of clauses, i.e., a set of sets of literals.

Definition 2.2.9. *A* truth assignment *assigns each variable x a value in* $\{\text{true}, \text{false}\}$.

This extends to literals and formulas in the standard way. Let val assign truth values to literals. Then

$$\begin{aligned} \text{val}(\neg x) &= \neg\, \text{val}(x), \\ \text{val}(x \wedge y) &= \text{val}(x) \wedge \text{val}(y), \\ \text{val}(x \vee y) &= \text{val}(x) \vee \text{val}(y). \end{aligned}$$

Definition 2.2.10. *A truth assignment that makes a formula true is called a* model *of the formula.*

2.2.1.1 Implication as a clause

Although CNF formulas are written using $\wedge, \vee, \neg$, it is extremely useful to use additional logical notation as shorthand that can be converted into CNF.

Let A and B be formulas. The implication $A \to B$ is logically equivalent to

$$\neg A \vee B,$$

since it is false exactly when A is true and B is false.

Suppose

$$A = a_1 \wedge a_2 \wedge \cdots \wedge a_r \quad \text{and} \quad B = b_1 \vee b_2 \vee \cdots \vee b_s,$$

where all a_i and b_j are literals. Then $A \to B$ is logically equivalent to

$$\neg(a_1 \wedge \cdots \wedge a_r) \vee (b_1 \vee \cdots \vee b_s).$$

By De Morgan's law,

$$\neg(a_1 \wedge \cdots \wedge a_r) \text{ is logically equivalent to } (\neg a_1) \vee (\neg a_2) \vee \cdots \vee (\neg a_r),$$

so

$$A \to B \text{ is logically equivalent to } \neg a_1 \vee \cdots \vee \neg a_r \vee b_1 \vee \cdots \vee b_s,$$

which is a clause. Therefore, whenever A is a conjunction of literals and B is a clause, the notation $A \to B$ can be regarded as shorthand for a single CNF clause.

We record a few special cases:

$$a \to b \text{ is logically equivalent to } \neg a \vee b,$$
$$(a_1 \wedge a_2) \to b \text{ is logically equivalent to } \neg a_1 \vee \neg a_2 \vee b,$$
$$(a_1 \wedge a_2) \to (b_1 \vee b_2) \text{ is logically equivalent to } \neg a_1 \vee \neg a_2 \vee b_1 \vee b_2.$$

2.2.1.2 Cardinality constraints

Encodings for Latin squares often require constraints about how many variables are true. Given literals $x_1, \ldots, x_t$, the clause

$$x_1 \vee x_2 \vee \cdots \vee x_t$$

enforces that at least one of the x_i is true.

To require that *at most one* of $x_1, \ldots, x_t$ is true, we forbid any pair from being simultaneously true: for all $1 \leq i < j \leq t$ we impose

$$\neg(x_i \wedge x_j) \text{ is logically equivalent to } \neg x_i \vee \neg x_j.$$

Thus an *at-most-one* constraint is encoded by

$$\bigwedge_{1 \leq i < j \leq t} (\neg x_i \vee \neg x_j).$$

Requiring *exactly one* of $x_1, \ldots, x_t$ to be true is achieved by combining at-least-one and at-most-one:

$$(x_1 \vee \cdots \vee x_t) \ \wedge \bigwedge_{1 \leq i < j \leq t} (\neg x_i \vee \neg x_j).$$

2.2.2 Encoding Latin squares

To encode a Latin square L of order n, we use Boolean variables

$$L_{r,c,s} \qquad (r, c, s \in \{0, 1, \ldots, n-1\}),$$

with the intended meaning

$$L_{r,c,s} = \text{true} \quad \Longleftrightarrow \quad L(r, c) = s.$$

The Latin square property is expressed by three families of *exactly-one* constraints.

2.2.2.1 Symbol constraints

Each cell has exactly one symbol. For every (r, c),

$$\bigvee_{s=0}^{n-1} L_{r,c,s} \qquad \text{and} \qquad \bigwedge_{0 \leq s < t \leq n-1} (\neg L_{r,c,s} \vee \neg L_{r,c,t}).$$

2.2.2.2 Row constraints

Each row contains each symbol exactly once. For every r and s,

$$\bigvee_{c=0}^{n-1} L_{r,c,s} \qquad \text{and} \qquad \bigwedge_{0 \le c < d \le n-1} (\neg L_{r,c,s} \vee \neg L_{r,d,s}).$$

2.2.2.3 Column constraints

Each column contains each symbol exactly once. For every c and s,

$$\bigvee_{r=0}^{n-1} L_{r,c,s} \qquad \text{and} \qquad \bigwedge_{0 \le r < u \le n-1} (\neg L_{r,c,s} \vee \neg L_{u,c,s}).$$

2.2.3 Orthogonality encoding for a set of 2 MOLS of order n

A Latin square of order n can be viewed as a function

$$L \colon \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}$$

such that, for each fixed row (resp. column), the map in the other coordinate is a permutation of $\{0, \ldots, n-1\}$. Two Latin squares L and L' are *orthogonal* if the map

$$(r, c) \longmapsto (L(r, c), L'(r, c))$$

is a bijection from $\{0, \ldots, n-1\}^2$ to $\{0, \ldots, n-1\}^2$. Equivalently, for every ordered pair $(a, b) \in \{0, \ldots, n-1\}^2$ there is a unique cell (r, c) with

$$(L(r, c), L'(r, c)) = (a, b).$$

In SAT, there are several standard ways to encode orthogonality. They differ primarily in the number of auxiliary variables introduced, the number and shape of clauses, and the rate of assignment of variables to their truth values (faster with more unit clauses), all of which can significantly impact solver performance.

2.2.3.1 Orthogonality by uniqueness of pairs

Let $L_{r,c,a}$ and $L'_{r,c,b}$ be variables encoding $L(r,c) = a$ and $L'(r,c) = b$, respectively. The Latin square constraints for L and L' are encoded as in Section 2.2.2.

A direct encoding of orthogonality forbids any ordered pair of symbols (a,b) from appearing in two distinct superimposed cells [123, Lemma 1]. Thus, for every (a,b) and every two distinct cells $(r,c) \neq (u,v)$, we add the clause

$$\neg L_{r,c,a} \ \vee \ \neg L'_{r,c,b} \ \vee \ \neg L_{u,v,a} \ \vee \ \neg L'_{u,v,b}. \tag{2.2.1}$$

This is equivalent to

$$(L_{r,c,a} \wedge L'_{r,c,b}) \ \rightarrow \ \neg(L_{u,v,a} \wedge L'_{u,v,b}),$$

and prevents the same ordered pair (a,b) from occurring in two different locations.

This encoding introduces no auxiliary variables, but it is large: it uses $\Theta(n^6)$ clauses of length 4.

2.2.3.2 Orthogonality by pairing variables

A common alternative introduces auxiliary variables

$$Z_{r,c,a,b} \qquad (r,c,a,b \in \{0,1,\ldots,n-1\}),$$

intended to mean

$$Z_{r,c,a,b} \leftrightarrow (L_{r,c,a} \wedge L'_{r,c,b}).$$

This equivalence can be enforced in CNF via Tseitin clauses [70, 111]:

$$(\neg Z_{r,c,a,b} \vee L_{r,c,a}), \quad (\neg Z_{r,c,a,b} \vee L'_{r,c,b}), \quad (\neg L_{r,c,a} \vee \neg L'_{r,c,b} \vee Z_{r,c,a,b}).$$

Orthogonality then becomes the requirement that, for each ordered pair (a,b), at most one of the variables $Z_{r,c,a,b}$ over all (r,c) is true (and hence, by counting with

the Latin constraints, exactly one is true). This makes the bijection in the definition of orthogonality explicit.

2.2.3.3 Orthogonality by composition of squares

A closely related orthogonality encoding, due to Zhang [123, Lemma 2], often performs well in practice. Although Zhang did not formulate the encoding in terms of a "composition square", their construction can be interpreted from this viewpoint. Our contribution is to make this composition perspective explicit, which also makes it natural to add further constraints that are not present in Zhang's [123, Lemma 2] original formulation.

View each row of a Latin square as a permutation of $\{0, \dots, n-1\}$. If L and L' are Latin squares of order n, define their *row-wise composition* LL' to be the array whose ith row is the permutation obtained by composing the ith row of L with the ith row of L' (right-to-left). In general, LL' need not be a Latin square. However, if L and L' are orthogonal, then LL'^{-1} is a Latin square [80].

In Chapter 4, we denote LL'^{-1} by Z and introduce variables to enforce that Z is a Latin square using the encoding from Section 2.2.2.

2.3 Group theory

We recall some basics from group theory. We will omit the group operation and write gh in place of $g \star h$ for $g, h \in G$ where the group operation is $\star$. Further, all groups in this dissertation are finite. The notion of groups will become useful in Chapter 5 where we use a special configuration of subgroups (defined below) of a group to construct Latin squares and sets of MOLS.

Definition 2.3.1 ([48]). *Let G be a group. A subset $H \subseteq G$ is a* subgroup *of G if H is nonempty and is closed under products and inverses (that is, if $x, y \in H$, then $x^{-1} \in H$ and $xy \in H$). If H is a subgroup of G, we write $H \leqslant G$.*

Definition 2.3.2 ([48]). *A subgroup $N \leq G$ is called a* normal subgroup *of G if it is*

invariant under conjugation by every element of G; that is, if

$$gNg^{-1} = N \quad \textit{for all } g \in G.$$

Equivalently, N is normal in G if for every $g \in G$ and every $n \in N$, we have $gng^{-1} \in N$. If N is a normal subgroup of G, we write

$$N \trianglelefteq G.$$

Let G be a group and let $H \leqslant G$ be a subgroup. For any $g \in G$, the set

$$gH \;=\; \{\, gh : h \in H \,\}$$

is a left coset of H in G. The index of H in G, denoted $[G : H]$, is the number of distinct left cosets of H in G. The set of left cosets is denoted by G/H.

Definition 2.3.3 ([48])**.** *Let G be a group and X a nonempty set. A (left)* action *of G on X is a map*

$$G \times X \to X, \qquad (g, x) \mapsto g \cdot x$$

such that for all $g, h \in G$ and $x \in X$,

$$e \cdot x = x, \qquad g \cdot (h \cdot x) = (gh) \cdot x,$$

where e is the identity element of G.

In this dissertation, we will use the term *group action* in place of *left group action.*

Definition 2.3.4 ([48])**.** *Let G act on X and let $x \in X$. The* orbit *of x under G, denoted* $\mathrm{Orb}_G(x)$*, is the set*

$$\mathrm{Orb}_G(x) := \{\, g \cdot x : g \in G \,\} \subseteq X,$$

and the stabilizer *of x is*

$$\mathrm{Stab}_G(x) := \{\, g \in G : g \cdot x = x \,\} \leq G.$$

The action is transitive *if it has a single orbit, i.e., for all $x, y \in X$ there exists $g \in G$ with $g \cdot x = y$. It is* simply transitive *if for all $x, y \in X$ there exists a* unique *$g \in G$ with $g \cdot x = y$. Equivalently, the action is simply transitive if it is transitive and $\mathrm{Stab}_G(x) = \{e\}$ for some (equivalently every) $x \in X$.*

It is important to note that a simply transitive action is sometimes called a *regular* or *sharply transitive* action in the group theory literature.

Definition 2.3.5 ([48])**.** *The* kernel *of the action is the set*

$$\{\, g \in G : g \cdot x = x \text{ for all } x \in X \,\},$$

which is a normal subgroup of G. The action is faithful *if the kernel is trivial.*

Theorem 2.3.6 (Orbit–stabilizer [48])**.** *Let G act on a finite set X and let $x \in X$. Then*

$$|\mathrm{Orb}_G(x)| = [G : \mathrm{Stab}_G(x)] = \frac{|G|}{|\mathrm{Stab}_G(x)|}.$$

In particular, the action is regular if and only if $|G| = |X|$ and the action is transitive.

Theorem 2.3.6 will be handy in Chapter 5.

2.4 Sets of MOLS based on a group

There are many ways to construct a set of MOLS—finite field construction for prime power orders, product construction, recursive and composition methods, and a variety of ad hoc families. While these methods provide broad existence theorems and asymptotic bounds, most of them have limited bearing on the central computational problems studied in this dissertation, which focus on structural restrictions and searches

in specific small orders (notably order 10). We therefore emphasize constructions where orthogonality can be expressed via explicit algebraic bijectivity conditions.

Constructions based on groups are tightly connected to MacNeish's conjecture: they furnish prototypical examples supporting the prime power case and motivate conjectural multiplicative behaviour under direct products. We return to this perspective in Chapter 5. The remainder of this section lays the background for later chapters: we relate transversals to complete mappings, encode sets of MOLS via the first rows in difference matrices, and express orthogonality through orthomorphisms and their pairwise orthogonality conditions.

We begin with transversals because they provide the most direct combinatorial evidence of orthogonality. For Cayley tables, the existence of a transversal is equivalent to the existence of an orthogonal mate, and this condition translates naturally into algebra via complete mappings.

2.4.1 Transversals and complete mappings

Definition 2.4.1. *A set of MOLS is* based on the group G *if each Latin square in the set is based on* G*.*

Given a quasigroup $(Q, \cdot)$ with $Q = \{q_0, \dots, q_{n-1}\}$ and a transversal $T = \{(i, j_i)\}_{i=0}^{n-1}$, define a map

$$\theta_T \colon Q \to Q, \qquad \theta_T(q_i) = q_{j_i}.$$

Because a transversal uses each column exactly once, $(j_0, \dots, j_{n-1})$ is a permutation of $\{0, \dots, n-1\}$, hence θ_T is a bijection.

Now consider the Cayley table of Q as a Latin square defined by $L(i, j) = q_i \cdot q_j$. The entries in the cells of the transversal are

$$L(i, j_i) = q_i \cdot q_{j_i} = q_i \cdot \theta_T(q_i).$$

The requirement that these entries are all distinct is exactly the statement that the

map

$$f_{\theta_T}\colon Q \to Q, \qquad f_{\theta_T}(q) = q \cdot \theta_T(q),$$

is also a bijection.

Definition 2.4.2. *Let $(Q, \cdot)$ be a quasigroup. A bijection $\theta\colon Q \to Q$ is a* complete mapping *if the map*

$$f_\theta\colon Q \to Q, \qquad f_\theta(q) = q \cdot \theta(q),$$

is also a bijection.

Example 2.4.3. *Let $Q = \mathbb{Z}_3 = \{0, 1, 2\}$ under addition* (mod 3). *Its Cayley table is:*

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

.

The diagonal cells form a transversal $T = \{(0,0), (1,1), (2,2)\}$. Here θ_T is the identity map, and

$$f_{\theta_T}(q) = q + \theta_T(q) = q + q = 2q \pmod 3,$$

which is a bijection. Hence, the identity map is a complete mapping of $\mathbb{Z}_3$.

Transversals correspond to complete mappings; for Cayley tables, this is equivalent to the existence of an orthogonal mate [52]. For example, let L be the Latin square given as the Cayley table of $\mathbb{Z}_3$. We construct an orthogonal mate L' of L using the complete mapping f_{θ_T} by defining (working $\mathbb{Z}_3$ additively)

$$L'(r, c) = r + f_{\theta_T}(c) = r + 2c \pmod 3.$$

The orthogonal mate is:

$$L' = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{bmatrix}$$

and it can be observed that L' is a permutation of the columns of L via f_{θ_T}, i.e., the column map $c \mapsto 2c \pmod 3$.

Example 2.4.4. *Let $Q = \mathbb{Z}_4 = \{0, 1, 2, 3\}$ and let $\theta(q) = q$. Then θ is a bijection, but*

$$f_\theta(q) = q + \theta(q) = 2q \pmod 4$$

takes values $0, 2, 0, 2$, so it is not a bijection. Hence θ is not a complete mapping.

However, this does not by itself rule out the existence of other complete mappings. In fact, it is known that $\mathbb{Z}_4$ admits no complete mappings [51]. Equivalently, the cyclic Latin square of order 4 has no transversal [50, 51], and therefore has no orthogonal mate.

Definition 2.4.5. *A* bachelor square *is a Latin square with no orthogonal mate.*

Euler [50, 51] proved that cyclic Latin squares of even order (Cayley tables of cyclic groups of even order) have no orthogonal mate, so bachelor squares exist for every even order. More generally, bachelor squares exist for all orders $n \neq 1, 3$ [51, 116]. In contrast, the Cayley tables of groups of odd order always admit orthogonal mates [52].

2.4.2 Difference matrices and orthomorphisms

Many strong lower bounds for $N(n)$ come from constructions of sets of MOLS based on groups [45]. Let $G = \{g_0, \ldots, g_{n-1}\}$ be a group of order n, and let L be its Cayley table where $L(i, j) = g_i g_j$. Permuting the columns of L by a permutation π yields a new Latin square whose first row is $(g_{\pi(0)}, \ldots, g_{\pi(n-1)})$, and the entire square is determined by that first row. Thus, sets of MOLS obtained from a Cayley table by column permutations can be encoded compactly by their first rows.

Definition 2.4.6. *Let G be a group of order n. An* (n, r)-difference matrix over G *is an $r \times n$ matrix $D = \{d_{ij}\}$ with entries in G such that for every pair of distinct rows $i \neq k$, the multiset of* row quotients

$$\{\, d_{ij}^{-1} d_{kj} : j = 1, \ldots, n \,\}$$

contains each element of G exactly once.

Example 2.4.7. *In the additive group $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$, the matrix*

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 4 & 1 & 3 \\ 0 & 3 & 1 & 4 & 2 \\ 0 & 4 & 3 & 2 & 1 \end{bmatrix}$$

is a $(5, 4)$-difference matrix: for any two distinct rows, the entrywise difference (mod 5) runs through all elements exactly once. The four rows are exactly the first rows of a set of 4 MOLS of order 5 below:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{bmatrix} \quad \begin{bmatrix} 0 & 2 & 4 & 1 & 3 \\ 1 & 3 & 0 & 2 & 4 \\ 2 & 4 & 1 & 3 & 0 \\ 3 & 0 & 2 & 4 & 1 \\ 4 & 1 & 3 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 3 & 1 & 4 & 2 \\ 1 & 4 & 2 & 0 & 3 \\ 2 & 0 & 3 & 1 & 4 \\ 3 & 1 & 4 & 2 & 0 \\ 4 & 2 & 0 & 3 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 4 & 3 & 2 & 1 \\ 1 & 0 & 4 & 3 & 2 \\ 2 & 1 & 0 & 4 & 3 \\ 3 & 2 & 1 & 0 & 4 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}.$$

There is an equivalent description using maps $G \to G$. Write G multiplicatively. Given a mapping $\theta\colon G \to G$, define an array L_θ by

$$L_\theta(i, j) = g_i\, \theta(g_j).$$

If θ is a bijection, then L_θ is obtained from the Cayley table by permuting columns, hence L_θ is a Latin square. Moreover, L_θ is orthogonal to L if and only if the map $x \mapsto x^{-1}\theta(x)$ is a bijection. More generally, if ϕ is another bijection, then L_θ is orthogonal to L_ϕ if and only if $x \mapsto \phi(x)^{-1}\theta(x)$ is a bijection.

Definition 2.4.8. *A bijection $\theta\colon G \to G$ is an* orthomorphism *of G if the map*

$$x \mapsto x^{-1}\theta(x)$$

is also a bijection.

Definition 2.4.9. *Two orthomorphisms θ and ϕ are* orthogonal *if the map*

$$x \mapsto \phi(x)^{-1}\theta(x)$$

is a bijection $G \to G$.

Example 2.4.10. *Let $G = \mathbb{Z}_5$ under addition. Then $x^{-1}\theta(x)$ becomes*

$$-x + \theta(x) = \theta(x) - x,$$

and $\phi(x)^{-1}\theta(x)$ becomes $\theta(x) - \phi(x)$.

For $a \in \mathbb{Z}_5$, let $\theta_a(x) = ax \pmod 5$. Then:

- *θ_a is a bijection if and only if $a \not\equiv 0 \pmod 5$.*
  - *If θ_a is a bijection, then it is an orthomorphism if and only if $(a-1)x$ is a bijection, equivalently $a \not\equiv 0, 1 \pmod 5$.*

Thus $\theta_2, \theta_3, \theta_4$ are orthomorphisms. For $a \neq b$,

$$\theta_a(x) - \theta_b(x) = (a-b)x$$

is bijective, so these orthomorphisms are pairwise orthogonal. The corresponding Latin squares are

$$L_{\theta_a}(i,j) = i + \theta_a(j) = i + aj \pmod 5,$$

yielding a set of 4 MOLS of order 5:

$$L,\ L_{\theta_2},\ L_{\theta_3},\ L_{\theta_4}.$$

The explicit Latin squares are given in Example 2.4.7.

Remark 2.4.11 ([52])**.** *Let G be a group of order n. The following are equivalent.*

1. *There exists a set of r pairwise orthogonal orthomorphisms of G.*

2. *There exists a set of $r+1$ MOLS, based on G, obtained by permuting the columns of the Cayley table of G.*

3. *There exists an $(n, r+2)$-difference matrix over G.*

Chapters 3–5 are incorporated verbatim , with some minor changes. Next, we declare the common infrastructure that underlies all the three sets of research results, and to isolate what is specific to each chapter.

2.5 Infrastructure

In Chapters 3 and 4, we use SAT solvers. Further, we used

- very similar functions that generate SAT instances from Latin square constraints, net constraints, symmetry-breaking constraints, and relation constraints from the 4-net viewpoint. However, the two chapters have different codebases;
- a modern CDCL SAT solver with DRAT-style proof logging enabled when unsatisfiability must be certified; and
- canonical labelling and equivalence testing via a fixed graph reduction and canonical certificate (nauty/SageMath), used to classify solutions up to the relevant equivalence.

When we report counts or classifications, they are always understood to be computed after applying this common notion of equivalence via canonical certificates.

In Chapters 5, we use GAP and SageMath. We searched for small groups in GAP's small group package and construct a subgroup configuration to build Latin squares from cosets of these subgroups and classify them up to main class equivalence.

Each chapter then adds its own problem-specific layer to common infrastructure or computational step: Chapter 3 specializes the SAT encoding and symmetry-breaking to the TRP/Myrvold framework and the extension-to-a-third-square problem; Chapter 4 specializes to the two-relation regime in the binary code of the associated 4-nets and emphasizes certified (UNSAT/DRAT-checked) case splits; and Chapter 5 is primarily

structural, using groups (no usage of a SAT solver) with computation serving to generate examples and to validate subgroup/configuration searches when needed.

CHAPTER 3

A 4×4 *subsquare in a set of* 3 MOLS(10)

We revisit a particularly structured subproblem studied by Myrvold [89]: orthogonal triples of order 10 in which one square contains a 4×4 Latin subsquare. Myrvold's analysis yields the necessary conditions in this setting and reduces the search for such triples to 28 orthogonal pair types. Using only constraints that involve two squares at a time, Myrvold eliminated 20 of these 28 types, leaving 8 cases.

Our main computational finding is that these remaining 8 cases cannot be eliminated by any strategy that only provides reasons for the existence of a suitable orthogonal pair. Indeed, for each of the remaining 8 pair types, we explicitly construct a transversal representation pair (TRP), and hence an orthogonal pair, satisfying the Myrvold framework. This explains, for the first time, why Myrvold's approach necessarily stops at eight cases: the obstruction is genuinely three-square in nature, rather than a deficiency of the two-square constraints.

Our contributions are as follows.

1. We implement a SAT encoding that quickly reproduces all Myrvold's 20 nonexistence results (two-square eliminations), providing an independent computational verification.

2. For each of the 8 remaining pair types, we find explicit TRPs (equivalently orthogonal pairs), thereby showing that these cases cannot be eliminated by focusing only on the existence of a suitable pair.

3. We give a concise formulation of the equivalence between orthogonality and transversal representations via a column-wise composition operation on column-Latin squares, which is different from row-wise composition operation in Laywine and Mullen [76]. This viewpoint yields an efficient SAT encoding for TRPs that performs well in practice.

This chapter reproduces *Myrvold's Results on Orthogonal Triples of* 10×10 *Latin Squares: A SAT Investigation*, co-authored with Curtis Bright and Brett Stevens, and published with the Electronic Journal of Combinatorics [34], and it is included verbatim, with some formatting changes to ensure consistency, as part of this dissertation.

3.1 Introduction

There are many ways of representing a transversal, but we follow Myrvold [89] and represent a transversal by listing the symbols in the transversal in each column from left to right. For example, the highlighted transversal in $\begin{bmatrix} 0 & \mathbf{1} & 2 \\ \mathbf{2} & 0 & 1 \\ 1 & 2 & \mathbf{0} \end{bmatrix}$ 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 is 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 [6, 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 $N(6) = 1$ [106]. However, in 1959, Euler's conjecture was shown to be false by the discovery of a 2 MOLS(22) [24] and a 2 MOLS(10) [95]. In fact, in 1960 it was shown that $N(n) \geq 2$ for all $n > 6$ [26]. 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 [82] 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 [96] proved that two orthogonal 10×10 Latin squares with orthogonal 3×3 Latin subsquares cannot be part of an orthogonal triple. Myrvold [89] 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.* [89]

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 3.2.

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 3.4 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 [89]. 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 3.3). 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 3.5 for more details.

3.2 Background

We define the notion of transversal representation and relate it to the orthogonality of Latin squares in Section 3.2.1. Next, we explain the transversal representation types classified by Myrvold [89] in Section 3.2.2, and give a brief description of satisfiability solving in Section 3.2.3. Lastly, we give a summary of related work in Section 3.2.4, with a focus on work applying automated reasoning tools to solve problems related to Latin squares.

3.2.1 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 [114]. From the n disjoint transversals,

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

Fig. 3.2.1: 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'.

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 3.2.1. 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 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 [76]. 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 3.3). 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 3.2.2 shows an example of this generalization; note the generalized transversals highlighted in D_1 contain duplicate entries and therefore are not traditional transversals. However,

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

Fig. 3.2.2: 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.

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.

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 $[a_0, \ldots, a_{n-1}]$ 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 jth 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 3.2.1. *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 3.2.1 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 3.2.2. *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 3.2.2 also implies the equality of i and i' because A and B are column-Latin squares. The consequent in Definition 3.2.2 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 3.2.1 and 3.2.2.

3.2.2 Transversal representation types

We now review Myrvold's results [89] 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 $\{0, 1, 2, \ldots, 9\}$ 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 3.2.3. 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 $\{4, 5, 6, \ldots, 9\}$ and each row and column of Σ must contain exactly $6 - 4 = 2$ symbols from the set $\{4, 5, 6, \ldots, 9\}$.

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 3.2.3). Since any transversal will contain exactly four white cells in total, it must contain $4 - i$ white cells in its first six

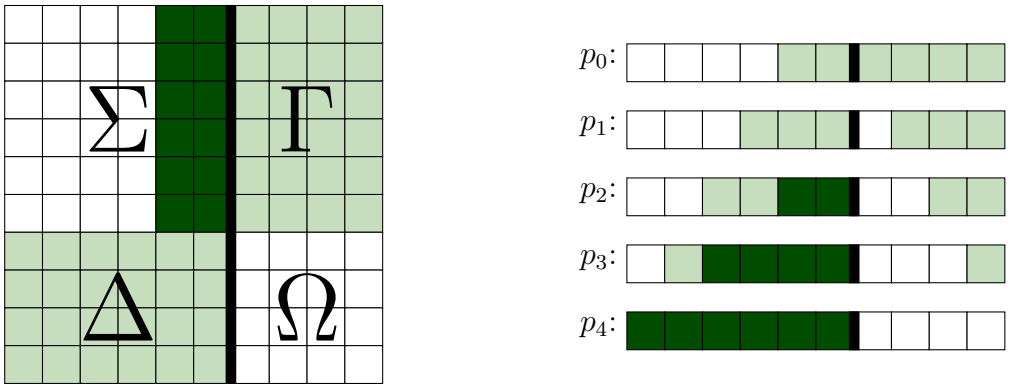


Fig. 3.2.3: The Latin square L (left) and its possible transversal types (right). White cells represent symbols in $\{0, 1, 2, 3\}$, light cells represent symbols in the rectangles Δ and Γ, and dark cells represent the symbols $\{4, 5, \ldots, 9\}$ in Σ. The cells of Σ are not shown in absolute positions; 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.

columns. Consider the entries of p_i that were chosen from the first six rows of L (i.e., Σ or Γ). We have $4 - i$ 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 3.2.3 ([89, Lemma 3.1])**.** *A transversal of type p_i contains exactly $2i - 2$ dark entries.*

A simple corollary of Lemma 3.2.3 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

Table 3.2.1: A summary of Myrvold's seven possible transversal types of L.

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

Diophantine linear system.

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

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 3.2.1 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 representation pair combinations. Under the assumption that L is part of an orthogonal triple, Myrvold [89, 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).

3.2.3 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 $l_1, \ldots, l_k$ 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 $c_1, \ldots, c_k$ are clauses.

When A is a conjunction of literals and B is a disjunction of literals, we use the notation $A \to B$ 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 $A \to B$ 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 [112], including mathematical problems such as the Boolean Pythagorean triples problem [59] and Lam's problem of proving the nonexistence of a projective plane of order ten [30]. Although these problems at first seem unconnected to logic, they can be reduced to SAT due to the versatility of Boolean logic [32]. 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 [74].

Lam's problem was resolved in 1989 using a massive computer search by Lam, Thiel, and Swiercz [75]. In 2011, the search was independently performed by Roy [100].

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. [30] 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 [100] was based in part on the nonexistence of a partial projective plane later determined to actually exist [28]. 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.

3.2.4 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 [85]. Independent computer searches [30, 75, 100] 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) [21, 88], these searches imply that no 9 MOLS(10)s exist or equivalently that $N(10) < 9$. Together with a result of Bruck [38], this implies that $N(10) \leq 6$ which is currently the best upper bound known on $N(10)$.

Egan and Wanless [49] 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 [16, 17] 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 $2\,\mathrm{MOLS}(n)$ for $n \leq 12$ and $3\,\mathrm{MOLS}(n)$ for $n \leq 9$. Rubin et al. [101] formulated a symmetry breaking method and also provided an alternative constraint programming encoding based on a theorem of Mann [80] 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 [78] use a general-purpose model searching program to find MOLS. They show a $k\,\mathrm{MOLS}(n)$ exists if and only if there exists a Latin square of order n which has $k-1$ transversal matrices $T_1, \ldots, T_{k-1}$ with any two transversal matrices T_i and T_j $(i \neq j)$ being transversal matrices of each other [78, Prop 1]. As a result, instead of searching for $k\,\mathrm{MOLS}(n)$, they searched for k Latin squares $L, T_1, \ldots, T_{k-1}$ that are mutual transversal matrices of each other. The initial Latin square L was defined as a function $f\colon \mathcal{R} \times \mathcal{C} \to \mathcal{D}$ on row indices $\mathcal{R}$, column indices $\mathcal{C}$, and symbol set $\mathcal{D}$. Similarly, the ith transversal matrix T_i $(1 \leq i \leq k-1)$ was defined as a function $f_i\colon \mathcal{D}_i \times \mathcal{C} \to \mathcal{R}$, 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 $k\,\mathrm{MOLS}(n)$ then consist of three types:

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

$$f(x_1, y) = f(x_2, y) \to x_1 = x_2, \qquad f(x, y_1) = f(x, y_2) \to y_1 = y_2,$$
$$f_i(t_1, y) = f_i(t_2, y) \to t_1 = t_2, \qquad f_i(t, y_1) = f_i(t, y_2) \to y_1 = y_2.$$

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

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

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:

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

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. [77] 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

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

A *Costas array* of order n is an $n \times n$ grid with n dots and $n^2 - n$ empty cells,

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

Fig. 3.2.4: An example 4×4 Costas Latin square.

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 3.2.4 for an example. Jin et al. [64] 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 $[0, \ldots, n-1]$ and replacing each column with its inverse [68]. In the rest of this chapter, we will use the notation A^{-1} for the column inverse of A (see Section 3.3.1).

A Latin square of order n is *idempotent* when its diagonal consists of the entries 0, 1, ..., $n-1$ 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. [62] 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 [121] 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 [122] 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 chapter relying on a consequence of Mann's theorem described in Section 3.3.1.

In a separate recently published investigation [33], we use a SAT encoding for orthogonality based on the one described in Section 3.4.2 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 [46] and was motivated by work of Dukes and Howard [47] 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 [58], 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.

3.3 Composition and duality

In this section, we describe a duality between the concepts of orthogonality and transversal representation. First, in Section 3.3.1 we define a composition operation on column-Latin squares. Then in Section 3.3.2 we use the composition operation to concisely characterize the duality.

3.3.1 Composition of column-Latin squares

A column-Latin square of order n can be represented by $(c_0, c_1, \ldots, c_{n-1})$ where c_j is the permutation of $[0, \ldots, n-1]$ formed by the jth 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 $F = (f_0, \ldots, f_{n-1})$ and $G = (g_0, \ldots, g_{n-1})$ is defined as

$$FG = (f_0 g_0, \ldots, f_{n-1} g_{n-1}).$$

The (i, j)th entry of FG is then $f_j g_j(i) = F[G[i, j], j]$. 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

$E = (e, \ldots, e)$ the column-Latin square of order n formed by n copies of e. The following two lemmas appear in Laywine and Mullen [76, pp. 98–99], except stated in terms of row-Latin squares instead of column-Latin squares.

Lemma 3.3.1. *Let C be a column-Latin square. Then (C, E) is an orthogonal pair if and only if C is a Latin square.*

Lemma 3.3.2. *If $\{C_1, C_2, \ldots, C_m\}$ is a set of mutually orthogonal column-Latin squares, then for any column-Latin square G, the set $\{C_1G, C_2G, \ldots, C_mG\}$ 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 [80] and also appears in Norton [91, Thm. 2] and Laywine–Mullin [76, 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 3.3.3. *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 3.3.1, (Z, E) is an orthogonal pair. By Lemma 3.3.2, (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 3.3.2, (Z, E) is an orthogonal pair (since $FC^{-1} = Z$ and $CC^{-1} = E$). By Lemma 3.3.1, 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, $j' \in \{0, 1, 2, \ldots, n-1\}$ where $j \neq j'$ with

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

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. □

3.3.2 Orthogonal pair / transversal representation duality

We now state a duality between orthogonality and transversal representations. This duality was already used by Myrvold [89, 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 3.3.4 and 3.3.5 are the analogue of Lemmas 3.3.1 and 3.3.2 with "orthogonal pair" replaced by "transversal representation pair".

Again, we let e denote the identity column permutation with $e(i) = i$ for $0 \leq i < n$ and $E = (e, \ldots, e)$ the column-Latin square of order n formed by n copies of e.

Lemma 3.3.4. *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. □

Lemma 3.3.5. *Let $\{C_1, C_2, \ldots, C_m\}$ be a set of mutual TRPs of column-Latin squares, then for any column-Latin square G, the set $\{GC_1, GC_2, \ldots, GC_m\}$ 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 (GC_1, GC_2) is not a TRP. Suppose there exist i, i', j, $j' \in \{0, 1, 2, \ldots, n-1\}$ 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. □

Proposition 3.3.6. *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 3.3.4, (Z, E) is a TRP. By Lemma 3.3.5, (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 3.3.5, (E, Z) is a TRP. Thus (E, Z) is a TRP. We have that Z is a Latin square by Lemma 3.3.4.

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, $j' \in \{0, 1, 2, \ldots, n-1\}$ 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. □

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 3.3.7 (cf. [89]). *Let $\mathcal{C}$ denote a set $\{C_1, \ldots, C_r\}$ of r column-Latin squares of order n.*

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

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

contains mutual TRPs.

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

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

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 3.3.3 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 3.3.3.

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 3.3.6 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 3.3.6. □

3.4 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 3.2.2. 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 3.4.1), are a transversal representation pair (see Section 3.4.2), and conform to one of Myrvold's 28 types (see Section 3.4.3).

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 3.4.4). 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 3.4.5). 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 3.4.6). Our encoding scripts are written in Python and are freely available at doi.org/10.5281/zenodo.18130631.

3.4.1 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 [123]. The direct method asserts that every cell of P contains *at least one* symbol and *at most one* symbol, i.e.,

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

Additionally, every column of P contains n distinct symbols,

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

and similarly every row of P contains n distinct symbols,

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

This encoding uses what is known as the binomial or pairwise encoding of the *exactly one* predicate [83] and uses $3n^2\left(\binom{n}{2} + 1\right)$ 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 [18]. 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_is as $\{0, 1\}$ variables) into conjunctive normal form. Using this encoding we specify that P is a Latin square with the cardinality constraints

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

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

3.4.2 Transversal representation constraints

The direct encoding that (P, Q) is a TRP using the contrapositive of Definition 3.2.1 would be

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

This is because if row i of P has its jth entry as k and its (j')th entry as k', then in whatever row of Q which has its jth 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 3.3.6 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 $0 \le i, j, k < n$) and Z is enforced to be a Latin square using the same encoding described in Section 3.4.1. 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

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

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

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

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 [123, 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*..."* [124]. 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

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

In light of the above and Proposition 3.3.3, 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

[1]The 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 chapter.

the efficiency of the solver in our experiments.

3.4.3 Colour constraints

We now describe how we encode that the square P is one of Myrvold's eight types described in Table 3.2.1; 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 `w` and `d` represent fixed symbols that are not in our symbol set $\{0, \ldots, n-1\}$.

We let the Boolean variable $P_{i,j,\mathtt{w}}$ represent that the (i,j)th entry of P is white, and let the Boolean variable $P_{i,j,\mathtt{d}}$ represent that the (i,j)th entry of P is dark. Otherwise, if both $P_{i,j,\mathtt{w}}$ and $P_{i,j,\mathtt{d}}$ 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 3.2.3). 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 $\{4, \ldots, 9\}$ not darkly coloured—making the variables $P_{i,j,\mathtt{w}}$ only necessary for $j \geq 6$. Altogether, we introduce n^2 new variables encoding the colours of P.

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

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

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

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

Recall that a transversal is said to be of type p_k when it has k whites in its last four entries. By Lemma 3.2.3, 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

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

Here, like in Section 3.4.1, we think of Boolean variables as taking $\{0, 1\}$ values and encode the cardinality constraints with the encoding of Bailleux and Boufkhad [18]. We also know that each of the first six columns of P contain exactly two dark entries, so we use the cardinality constraints

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

Similarly, we also use n^2 Boolean variables $Q_{i,j,\mathtt{w}}$ and $Q_{i,j,\mathtt{d}}$ to represent the colours of the square Q and add similar constraints to those above (using the $Q_{i,j,\mathtt{w}}$ and $Q_{i,j,\mathtt{d}}$ variables in place of the $P_{i,j,\mathtt{w}}$ and $P_{i,j,\mathtt{d}}$ 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, ..., 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 jth column having symbol k must also be coloured dark. The symbol k must exist in the jth 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

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

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

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

3.4.4 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 3.2.2, 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 [85]. 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

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

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

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

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

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

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.

3.4.5 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 $\{4,\ldots,9\}$ 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 $\{\text{white}, \text{light}, \text{dark}\}$ corresponding to Myrvold's types from Section 3.2.2. If (P,L) is a coloured TRP where L has been coloured corresponding to Figure 3.2.3, 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 $\{4,\ldots,9\}$ 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 3.4.1. *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*

0	1	2	4	5	6	3	7	8	9

,

0	1	3	4	5	6	2	7	8	9

, *or*

0	2	3	4	5	6	1	7	8	9

.

In Theorem 3.4.3, 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 3.4.3.

Lemma 3.4.2. *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) [85]. The two additional reduced Latin squares of

order four are both isotopic to Ω_1 and are given by

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

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. □

Theorem 3.4.3. *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 3.2.3. 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 3.4.2, 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 $\{4, \ldots, 9\}$ 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 3.4.1.

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. □

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,

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

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

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

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

Enforcing the fact that rows are sorted by transversal type is done with the cardinality constraints discussed in Section 3.4.3, 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 $P_{i,6,\mathtt{w}} + \cdots + P_{i,9,\mathtt{w}} = 1$ and $P_{i,0,\mathtt{d}} + \cdots + P_{i,5,\mathtt{d}} = 0$ for $0 \leq i < 8$, and the last two rows of P to be of type p_4 with $P_{i,6,\mathtt{w}} + \cdots + P_{i,9,\mathtt{w}} = 4$ and $P_{i,0,\mathtt{d}} + \cdots + P_{i,5,\mathtt{d}} = 6$ 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.

3.4.6 Postprocessing

As we will describe in Section 3.5, 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 3.4.2 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 [49], and finally checking the graphs for equivalence using the graph isomorphism tool NAUTY [86].

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 $\{0, \ldots, n-1\}$, 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.

3.5 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 [20] run on AMD EPYC Zen 5 processors running at 2.7 GHz and equipped with 1 GiB of memory.

Recall Myrvold showed [89, 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 Section 3.5.1 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 3.5.1, and a scatterplot of the running times is given in Figure 3.5.1. 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 3.5.2. In particular, for each pair type we provide the number of TRPs found that are

Table 3.5.1: 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	mean	median	min	max
(U, U)	031102.1	019619.4	0748.6	098009.2
(S, X)	058780.5	038453.2	2282.4	175005.1
(U, W)	075043.9	056171.4	2659.8	399428.7
(W, W)	139198.5	097661.6	1662.1	timeout
(V, X)	147169.2	114191.0	2567.7	timeout
(U, X)	140560.4	117378.6	0327.8	timeout
(W, X)	222515.7	176970.4	0527.3	timeout
(X, X)	429809.6	580524.5	6747.1	timeout

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 Section 3.5.1. Also listed in Table 3.5.2 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.

Table 3.5.2: 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.

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

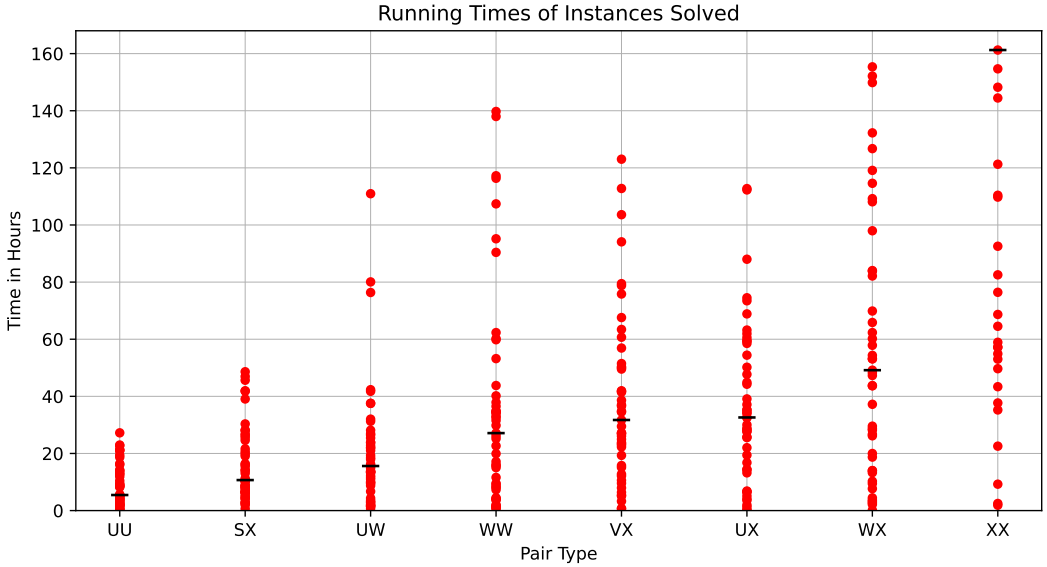


Fig. 3.5.1: A scatterplot of the solver's running time for each pair type. The median running time is shown as a solid black line. Timeouts are not plotted but are used in determining the median.

3.5.1 Examples of the design

We provide eight explicit pairs we found which prove the existence of TRPs for Myrvold's eight unresolved cases [89].

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

3.6 Conclusion

In this chapter we use a satisfiability (SAT) solver to investigate Myrvold's nonexistence results [89] 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 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.

CHAPTER 4

Enumeration of 4*-nets of order* 10 *with two relations*

A 4-net(10) is equivalent to an orthogonal pair of Latin squares of order 10. A central invariant of a net is the rank of its incidence matrix over $\mathbb{F}_2$. Nontrivial linear dependencies among the columns of this matrix are called relations. The case of order 10 is especially compelling: it lies at the boundary where the maximum number of mutually orthogonal Latin squares is unknown, and structural restrictions on orthogonal pairs can provide meaningful obstructions to extending candidates toward a set of 3 MOLS(10).

Howard [60] gave an explicit outline of what a relation in 4-net(10) would entail. Delisle [46] carried out an exhaustive enumeration of all orthogonal pairs of order 10 whose corresponding 4-nets have at least two nontrivial relations. Myrvold independently verified their enumeration. We revisited their classification using a SAT-based approach rather than custom backtracking code. Our guiding principle is that the defining properties of the objects under study—Latin squares, orthogonality, relation constraints, and symmetry breaking—can be expressed as a Boolean formula in conjunctive normal form. Modern SAT solvers then provide an effective search engine for exhaustive enumeration and can also emit certificates of unsatisfiability, which are independently checkable by proof verifiers.

Our contributions are as follows.

1. A SAT encoding is given for orthogonal pairs of Latin squares of order 10 whose associated 4-nets have at least two nontrivial relations, incorporating the

symmetry-breaking framework used in Delisle's search.

2. Using IPASIR-UP functionality, the SAT solver is adapted to perform exhaustive enumeration by injecting blocking clauses, yielding a complete list of solutions and final unsatisfiability certificates once all solutions have been blocked.

3. The resulting enumeration confirms the earlier classifications: out of five cases, three are shown to be unsatisfiable, and two produce examples with the solution counts and main class totals matching the earlier works (including identification of a misprint in the previously reported count for one of the five cases).

4. A short counting argument (separate from SAT proof certificates) is provided to explain the impossibility of one of the unsatisfiable cases.

This chapter reproduces *Orthogonal Latin Squares of Order Ten with Two Relations: A SAT Investigation*, co-authored with Curtis Bright and Brett Stevens, and published with the journal of Discrete Mathematics, Algorithms and Applications [33], and it is included verbatim, with some formatting changes to ensure consistency, as part of this dissertation.

4.1 Introduction

In this chapter without loss of generality we assume that the lines in a net are always ordered by parallel class, i.e., the first parallel class consists of the first n lines in the net, the second parallel class consists of the next n lines in the net, etc. Under this ordering, the axioms of a net imply that if A is the incidence matrix of a k-net(n), then

$$A^T A = \begin{bmatrix} nI & J & \cdots & J \\ J & nI & \cdots & J \\ \vdots & \vdots & \ddots & \vdots \\ J & J & \cdots & nI \end{bmatrix} \tag{4.1.1}$$

where here A and A^T are considered as matrices over $\mathbb{Z}$, I is the identity matrix of order n, and J is the all-ones matrix of order n. This follows because the (i, j)th entry

of A^TA counts the number of points occurring on both the ith and jth line, and this count is 1 when the lines are in different parallel classes, 0 when the lines are distinct and in the same parallel class, and n when the lines are identical (i.e., $i = j$).

It is well-known that a k-net(n) is equivalent to $k-2$ mutually orthogonal Latin squares of order n, denoted $(k-2)\,\mathrm{MOLS}(n)$, and it is straightforward to convert a k-net to a collection of $k-2$ Latin squares and vice versa. Roughly speaking, the net's first parallel class corresponds to Latin square rows, the net's second parallel class corresponds to Latin square columns, and for $\ell > 2$ the net's ℓth parallel class corresponds to symbols of the $(\ell-2)$th Latin square. When the (i,j)th entry of the $(\ell-2)$th Latin square contains symbol s, its corresponding net will contain a point that lies on the ith line of the first parallel class, the jth line of the second parallel class, and the sth line of the ℓth parallel class.

A *relation* in a k-net(n) is a linear dependency in the columns of A over $\mathbb{F}_2$. The *rank* of k-net(n) is the rank of its incidence matrix. The rank of a k-net(n) is at most $kn - k + 1$ (c.f. [61]) since there are $k-1$ trivial relations formed by the lines in the first parallel class and the ith parallel class for $2 \leq i \leq k$. Howard [60] considered nets of order $n \equiv 2 \pmod 4$ of which the case $n = 10$ is of particular interest. She showed that a 6-net(10) has rank ≤ 53 and therefore contains at least two nontrivial relations, and also that a 4-net(10) has rank ≥ 33 and therefore contains at most four nontrivial relations.

Delisle, under the supervision of Myrvold, computationally enumerated all 4-nets(10) with at least two nontrivial relations [46]. They found that there exist no 4-nets(10) with four nontrivial relations, up to isomorphism there are six 4-nets(10) with exactly three nontrivial relations, and up to isomorphism there are 85 4-nets(10) with exactly two nontrivial relations. More recently, Gill and Wanless [58] computationally enumerated all 4-nets(10) with at least one nontrivial relation and found that up to isomorphism there are exactly 18,526,229 4-nets(10) with exactly one nontrivial relation.

A 4-net(10) is equivalent to an orthogonal pair of Latin squares of order 10. Latin squares of order 10 are of particular interest because 10 is the first order for which the

largest collection of mutually orthogonal Latin squares is unknown. It is known that 9 mutually orthogonal Latin squares of order 10 would be equivalent to a projective plane of order ten, but this was ruled out by exhaustive computer search [75]. Combined with a result of Bruck [38], this implies that 7 MOLS(10) do not exist. Thus, the maximum number of mutually orthogonal Latin squares of order 10 is between 2 and 6. As a consequence of the searches of Delisle [46] and Gill and Wanless [58], all 2 MOLS(10) with nontrivial relations are known and none of them are part of a 3 MOLS(10).

In this chapter, we verify the searches of Delisle [46] by exhaustively enumerating all 2 MOLS(10) with two nontrivial relations. Our approach differs from Delisle's original search and the independent verification of Gill and Wanless [58] because our search uses a Boolean satisfiability (SAT) solver. We reduce the existence of a 2 MOLS(10) with two relations into a problem of Boolean logic and then use a SAT solver to find all solutions of the logic problem (and correspondingly all 2 MOLS(10) with two relations). SAT solvers can be surprisingly effective at solving various problems in mathematics [32] and recently they have been used with increasing frequency to solve problems in combinatorics and design theory [124]. For example, in 2021 they were used to verify Lam et al.'s result that order ten projective planes do not exist [30].

Traditionally, backtracking algorithms are used to computationally search for combinatorial designs [67]. SAT solvers offer an alternative approach to the backtracking paradigm. Although SAT solvers also perform a form of backtracking, they also use a powerful learning process known as conflict-driven clause-learning [84]. This technique along with many other optimizations and heuristics that have been fine-tuned over decades enables SAT solvers to outperform traditional backtracking search in many problems of interest.

Moreover, even if SAT solvers were not as efficient as custom backtracking approaches, they have the advantage of making the search process less error-prone because using a SAT solver does not require writing any code for performing a search. Unfortunately, it is a reality of software development that almost all computer programs have bugs, and writing efficient search code is an inherently error-prone process [73].

This is a particularly important consideration when a computer program purports to perform an exhaustive classification of a mathematical object. How can we trust that a bug in the program did not cause some objects to be missed in the search? It is typically impossible for even professional programmers to guarantee their code has no bugs due to the inherent difficulties in writing computer code. One way of decreasing the chance that a bug results in a missed object is to perform the same search with two implementations written independently. While this does reduce the chance of something being missed, it still relies on trusting code that cannot be certified to be correct. Indeed, it has happened that mathematical designs have been missed by a search with an independent verification, for example, in Lam's problem [35] and the enumeration of good matrices of order 27 [31].

Using a SAT solver sidesteps the need for writing search code. Instead, one generates a "SAT encoding" specifying the properties that the object in question must satisfy, and then the SAT solver searches for the object. Moreover, the SAT solver *does not need to be trusted itself*, because during the search it produces a certificate that can be checked for correctness by a proof verifier—a simpler piece of software that can be written independently from the SAT solver. This approach does rely on the SAT encoding being correct, but this typically requires less trust than would be required of a search procedure. We describe the SAT encoding we use in Section 4.3, a crucial component of which is a SAT encoding of the symmetry breaking used in Delisle's original search (described in detail along with other background in Section 4.2).

SAT solvers do not always perform well on mathematical problems, particularly when there is underlying structure in the problem unknown to the solver. As an example, integer factorization can be reduced to the SAT problem, but SAT solvers cannot factorize integers nearly as efficiently as algorithms specifically designed to do so using number theoretic concepts. Augmenting SAT solvers with number theoretic reasoning improves their performance, but even still they only outperform traditional algorithms when extra information is available such as random bits of the prime factors [12]. Despite SAT solvers having no efficiency guarantee, they did perform remarkably well in the enumeration of orthogonal pairs of Latin squares of order ten

with two relations. Only minor modifications to the SAT solver and proof verifier were required, as described in Section 4.4. We enumerated all 91 pairs of Latin squares of order ten with two relations in less than 1.75 hours on a desktop computer (see Section 4.5). Conversely, Delisle's original backtracking search, written in the programming language C and optimized for speed, used around 488 CPU days in 2010. While some of the improvements in speed of our SAT-based search is undoubtedly from the improvements in computer hardware, this alone cannot account for an over 6000× speedup in CPU time, demonstrating the computational efficiency of SAT solvers on problems of this type.

4.2 Background

The *type* of a relation is a list of the number of lines each parallel class contributes to the relation. For example, a relation in a 4-net(n) that has 4 lines from the first two parallel classes and 6 lines from the last two parallel classes has type $[4, 4, 6, 6]$. Because the union of two parallel classes form a trivial relation, without loss of generality any relation can be "complemented" and written in a form where at most one entry in the type is larger than $n/2$. In her PhD thesis, Howard [60] studies relations in nets of order $n \equiv 2 \pmod 4$. In particular, she proved the following proposition.

Proposition 4.2.1 ([60, Prop. 5.4])**.** *Every nonempty relation in a 4-net of order* $n \equiv 2 \pmod 4$ *with at most* $n/2$ *lines in three classes must be of the type* $[k, k, k, k]$ *where* k *is an even integer with* $n/3 \leq k < n/2$.

For example, Proposition 4.2.1 implies that every nontrivial relation in a 4-net(10) is complementable to one of type $[4, 4, 4, 4]$, because every nontrivial relation in a 4-net(10) is complementable to a nonempty relation with at most 5 lines in three classes. Now consider the case of a 4-net(10) with two linearly independent nontrivial relations. Note the sum of two nontrivial relations R_1 and R_2 will be a third relation, and because the relations are over $\mathbb{F}_2$, the sum is the symmetric difference $(R_1 \cup R_2) \setminus (R_1 \cap R_2)$. Although this third relation will not be linearly independent to R_1 and R_2, it will

be nontrivial, and therefore by Proposition 4.2.1 complementable to a relation of the type $[4,4,4,4]$. The following proposition constrains the manner in which two linearly independent relations intersect.

Proposition 4.2.2 (cf. [46])**.** *Suppose R_1 and R_2 are two linearly independent relations of type $[4,4,4,4]$ in a 4-net(10). Then each parallel class contains exactly one or two lines from both R_1 and R_2.*

Proof. Suppose x is the number of lines in the first parallel class and both R_1 and R_2. Since the sum of R_1 and R_2 is a third relation that is complementable to one of type $[4,4,4,4]$, we have that $(4-x)+(4-x)$, the number of lines in the third relation and the first parallel class, is either 4 or $10-4=6$. In the former case $x=2$ and in the latter case $x=1$. The same argument applies to every parallel class. □

Following Delisle, we will suppose the lines appearing in both R_1 and R_2 are ordered to appear first in each parallel class, followed by the lines in R_1 and not R_2, and then the lines in R_2 and not R_1. The remaining lines, those not in either R_1 and R_2, appear last. Say that there are x_i lines in parallel class i and $R_1 \cap R_2$, there are y_i lines in parallel class i and $R_1 \setminus R_2$, and there are z_i lines in parallel class i and $R_2 \setminus R_1$. We use the notation $[[x_1,y_1,z_1],[x_2,y_2,z_2],[x_3,y_3,z_3],[x_4,y_4,z_4]]$ to denote the form of R_1 and R_2. By Proposition 4.2.2, without loss of generality the possible forms can be taken to be one of the five cases

$$[[1,3,3],[1,3,3],[1,3,3],[1,3,3]], \qquad \text{(case 1)}$$

$$[[1,3,3],[1,3,3],[1,3,3],[2,2,2]], \qquad \text{(case 2)}$$

$$[[1,3,3],[1,3,3],[2,2,2],[2,2,2]], \qquad \text{(case 3)}$$

$$[[1,3,3],[2,2,2],[2,2,2],[2,2,2]], \qquad \text{(case 4)}$$

$$[[2,2,2],[2,2,2],[2,2,2],[2,2,2]]. \qquad \text{(case 5)}$$

Furthermore, Delisle used a counting argument to rule out cases 2 and 4 [46, pg. 17]. Thus, Delisle's search focused on cases 1, 3, and 5. Interestingly, using our approach a SAT solver rules out cases 2, 3, and 4 in a few seconds each. Given case 3 was ruled

out by Delisle using 23 days of compute time, the SAT solver shows its strength at uncovering contradictions by ruling out case 3 quite quickly. At the end of Section 4.5, we also give an argument using no search and a short computation that rules out case 3. We found this argument independently of the SAT computation.

4.2.1 Delisle's symmetry breaking

In order to perform an exhaustive enumeration up to isomorphism it is advantageous to restrict the search space as much as possible without losing any solutions up to isomorphism—this process is known as *symmetry breaking*. In this section we recount the symmetry breaking used in Delisle's thesis [46] for 4-nets(10) with two relations (stated in terms of 2 MOLS(10)). Say (A, B) is a 2 MOLS(10) whose corresponding net has 2 nontrivial relations. Delisle's method for adding symmetry breaking constraints on (A, B) is based on adding additional constraints on the entries in the first column and row of A and B.

Suppose (A, B) is a pair of orthogonal Latin squares with two relations of the form

$$[[x_1, y_1, z_1], [x_2, y_2, z_2], [x_3, y_3, z_3], [x_4, y_4, z_4]]. \tag{4.2.1}$$

The form of the relations define equivalence classes on the rows (from parallel class 1), columns (from parallel class 2), symbols of A (from parallel class 3), and symbols of B (from parallel class 4). For example, the row equivalence classes are determined by the values of x_1, y_1, and z_1: explicitly, the equivalence classes of rows will be defined by the index sets $[0, x_1)$, $[x_1, x_1 + y_1)$, $[x_1 + y_1, x_1 + y_1 + z_1)$, and $[x_1 + y_1 + z_1, 10)$. Note that the following six equivalence operations on pairs of Latin squares (A, B) preserve the orthogonality of the squares and the form of the relations.

1. Permutation of rows of A and B preserving row equivalence classes (the same permutation applied to A and B simultaneously).

2. Permutation of columns of A and B preserving column equivalence classes (the same permutation applied to A and B simultaneously).

3. Permutation of the symbols of A preserving A-symbol equivalence classes.

4. Permutation of the symbols of B preserving B-symbol equivalence classes.

5. Taking the transpose of A and B (when the row and column equivalence classes match, i.e., cases 1–3 and 5).

6. Swapping A and B (when the A-symbol and B-symbol equivalence classes match, i.e., cases 1 and 3–5).

Fix an ordering of the entries of a Latin square in the following way: the entries of the first column (from top to bottom) are first, and then the entries of the first row (from left to right) are next. The remaining entries can be ordered arbitrarily. In this way, if A and A' are two distinct Latin squares we say $A < A'$ if on the first entry in which A and A' differ, say at index (i, j), we have $A_{ij} < A'_{ij}$. Similarly, pairs of Latin squares can be ordered after providing an ordering on pairs of symbols. For this, a lexicographic ordering is used: say that $(a, b) < (a', b')$ when either $a < a'$ or $a = a'$ and $b < b'$. Then, if (A, B) and (A', B') are two distinct pairs of Latin squares, we say $(A, B) < (A', B')$ if on the first entry in which (A, B) and (A', B') differ, say at index (i, j), we have $(A_{ij}, B_{ij}) < (A'_{ij}, B'_{ij})$.

A pair of orthogonal Latin squares (A, B) is said to be a *minimal pair* if (A, B) cannot be decreased under the ordering described above by using the equivalence operations described above. Delisle's symmetry breaking method is based on the following six propositions. In each, (A, B) is a pair of orthogonal Latin squares and each proposition gives a necessary condition for (A, B) to be a minimal pair.

Proposition 4.2.3. *If (A, B) is a minimal pair then $(A_{1,0}, B_{1,0}) < (A_{0,1}, B_{0,1})$ (except possibly in case 4).*

Proof. Suppose (A, B) is a minimal pair with $(A_{1,0}, B_{1,0}) \geq (A_{0,1}, B_{0,1})$. Since A and B are orthogonal, $(A_{1,0}, B_{1,0})$ and $(A_{0,1}, B_{0,1})$ are distinct, and thus $(A_{1,0}, B_{1,0}) > (A_{0,1}, B_{0,1})$. Applying the transpose operation to A and B does not affect $A_{0,0}$ and $B_{0,0}$, but replaces $(A_{1,0}, B_{1,0})$ with $(A_{0,1}, B_{0,1})$, so $(A^T, B^T) < (A, B)$ in contradiction

to the fact that (A, B) is minimal. (This argument does not work in case 4, as the transpose operation is not an equivalence operation in case 4.) □

Proposition 4.2.4. *If (A, B) is a minimal pair then $A < B$ (except possibly in case 2).*

Proof. Suppose (A, B) is a minimal pair with $A \geq B$. Since A and B are orthogonal, $A \neq B$, and thus $A > B$ and there is some entry on which A and B do not match. After swapping A and B the first entry on which the mismatch occurs will still be in the same place, so $(B, A) < (A, B)$ in contradiction to (A, B) being minimal. (This argument does not work in case 2, as swapping A and B is not an equivalence operation in case 2.) □

Proposition 4.2.5. *If (A, B) is a minimal pair then the symbols in the first column of A appear in sorted order within the rows of each row equivalence class.*

Proof. Suppose (A, B) is a minimal pair with the symbols in the first column of A not in sorted order within the rows of each row equivalence class. Using row permutations, permute the rows of (A, B) to form (A', B') such that the rows of the first column of A' are now sorted within the rows of each row equivalence class. This is possible because we can freely permute rows of A and B if we focus only on the rows in one particular row equivalence class. Thus, there is some permutation which sorts those particular rows by the entries of the first column of A. Consider the first entry $A'_{i,0}$ of A' that has changed after applying these permutations. (This will also be the first entry of B' that has changed, since the same permutations are applied to A and B.) Because $A'_{i,0} < A_{i,0}$ and $(A'_{j,0}, B'_{j,0}) = (A_{j,0}, B_{j,0})$ for all $j < i$ we have $(A', B') < (A, B)$ in contradiction to (A, B) being minimal. □

Proposition 4.2.6. *If (A, B) is a minimal pair then the symbols in the first row of A appear in sorted order within the columns of each column equivalence class (except possibly in case 4; in that case $A_{0,0}$ and $A_{0,1}$ may appear out of order).*

Proof. Suppose (A, B) is a minimal pair with the symbols in the first row of A not in sorted order within the columns of each column equivalence class. First consider

cases 1–3, in which case the first column equivalence class consists of only the first column. In this case, the proposition is vacuous for the first column equivalence class, as the list $[A_{0,0}]$ has length 1 and is vacuously sorted. Using column permutations, permute the columns of (A, B) to form (A', B') such that the columns of the first row of A' are now sorted within the columns of the remaining three column equivalence classes. Note that the first column of (A', B') matches the first column of (A, B) since the first column was not permuted. Consider the first entry $A'_{0,i}$ that has changed after applying these permutations. Because $A'_{0,i} < A_{0,i}$ and $(0, i)$ is the first entry in which (A', B') differs from (A, B), it follows that $(A', B') < (A, B)$, in contradiction to (A, B) being minimal.

In cases 4 and 5, the above argument works to sort the entries in the first row of A in each of the last three column equivalence classes, but not the first, so $A_{0,0}$ and $A_{0,1}$ may be out of order. However, in case 5, since (A, B) is a minimal pair whose first two rows are in the same row equivalence class, by Proposition 4.2.5 we have $A_{0,0} < A_{1,0}$. By Proposition 4.2.3, we also have $A_{1,0} \leq A_{0,1}$. Thus in case 5 the entries of $A_{0,0}$ and $A_{0,1}$ will also appear in sorted order. □

Proposition 4.2.7. *If (A, B) is a minimal pair then for each A-symbol equivalence class, the symbols of that equivalence class in the first column of A appear in sorted order.*

Proof. Suppose (A, B) is a minimal pair where the symbols in the same A-symbol equivalence class in the first column of A do not appear in sorted order. Permute the symbols of A to form A' so that for each A-symbol equivalence class the symbols in that equivalence class in the first column of A' appear in sorted order. Consider the first entry $A'_{i,0}$ that differs from $A_{i,0}$. Since $A'_{i,0} < A_{i,0}$ and $A'_{j,0} = A_{j,0}$ for all $j < i$, we have $(A', B) < (A, B)$ in contradiction to (A, B) being minimal. □

Proposition 4.2.8. *If (A, B) is a minimal pair then for each B-symbol equivalence class, the symbols of that equivalence class in the first column of B appear in sorted order.*

Proof. Suppose (A, B) is a minimal pair where the symbols in the same B-symbol

equivalence class in the first column of B do not appear in sorted order. Permute the symbols of B to form B' so that for each B-symbol equivalence class the symbols in that equivalence class in the first column of B' appear in sorted order. Consider the first entry $B'_{i,0}$ that differs from $B_{i,0}$. Since $B'_{i,0} < B_{i,0}$ and $B'_{j,0} = B_{j,0}$ for all $j < i$, we have $(A, B') < (A, B)$ in contradiction to (A, B) being minimal. □

Finally, we prove another property of minimal pairs that we exploit in our encoding.

Proposition 4.2.9. *If (A, B) is a minimal pair then $A_{0,0} = B_{0,0}$ (except possibly in case 2).*

Proof. By Proposition 4.2.7 if (A, B) is a minimal pair then $A_{0,0}$ must be in $\{0, 1, 4, 7\}$ in case 1 and in $\{0, 2, 4, 6\}$ in cases 3–5. Similarly, by Proposition 4.2.8 if (A, B) is a minimal pair then $B_{0,0}$ must be in $\{0, 1, 4, 7\}$ in case 1 and in $\{0, 2, 4, 6\}$ in cases 3–5. Delisle [46, pg. 18] gives the possibilities for the values of $(A_{0,0}, B_{0,0})$ in case 1, and the only ones which are in $\{0, 1, 4, 7\} \times \{0, 1, 4, 7\}$ are $(0, 0)$, $(1, 1)$, $(4, 4)$, and $(7, 7)$. Similarly, [46, pg. 20] gives the possibilities for the values of $(A_{0,0}, B_{0,0})$ in case 3 (and these are identical in cases 4 and 5), and the only ones which are in $\{0, 2, 4, 6\} \times \{0, 2, 4, 6\}$ are $(0, 0)$, $(2, 2)$, $(4, 4)$, and $(6, 6)$. □

4.3 SAT encoding

In this section we describe our SAT encoding for the problem of enumerating 2 MOLS(10) with two nontrivial relations. In order to encode a pair of orthogonal Latin squares (A, B) of order n we use the $2n^3$ Boolean variables A_{ijk} and B_{ijk} for $0 \leq i < n$. The variable A_{ijk} will be true exactly when the (i, j)th entry of square A contains the symbol k, and similarly for the variables B_{ijk} and the entries of the square B.

Modern SAT solvers require the input formulae to be in a format known as conjunctive normal form. An expression in Boolean logic is in *conjunctive normal form* when it is a conjunction of clauses, a *clause* being a disjunction of variables or negated variables. For example, $\neg x \vee y \vee z$ is a clause. We may use the implication

operator to express clauses with the meaning that $(x_1 \wedge \cdots \wedge x_n) \to (y_1 \vee \cdots \vee y_m)$ is shorthand for the clause $\neg x_1 \vee \cdots \vee \neg x_n \vee y_1 \vee \cdots \vee y_m$.

Our SAT encoding contains four kinds of constraints: constraints asserting that A and B are Latin squares (see Section 4.3.1), constraints asserting that A and B are orthogonal (see Section 4.3.2), constraints asserting that A and B have two nontrivial relations and are in one of the forms specified by cases 1–5 (see Section 4.3.3), and finally symmetry breaking constraints asserting that A and B satisfy Propositions 4.2.3 to 4.2.8 (see Section 4.3.4).

4.3.1 Latin square encoding

Considering the Boolean variables A_{ijk} as integers (0 for false and 1 for true), in order to specify that they encode a Latin square of order 10 we need to enforce the following three constraints.

1. $\sum_{k=0}^{9} A_{ijk} = 1$ for all $0 \leq i, j \leq 9$ (every cell has exactly one symbol).
2. $\sum_{j=0}^{9} A_{ijk} = 1$ for all $0 \leq i, k \leq 9$ (every row contains every symbol exactly once).
3. $\sum_{i=0}^{9} A_{ijk} = 1$ for all $0 \leq j, k \leq 9$ (every column contains every symbol exactly once).

The most straightforward way of representing the constraint $\sum_{i=1}^{n} x_i = 1$ in Boolean logic is to encode $\sum_{i=1}^{n} x_i \leq 1$ via $\bigwedge_{i<j}(\neg x_i \vee \neg x_j)$ and $\sum_{i=1}^{n} x_i \geq 1$ via $\bigvee_{i=1}^{n} x_i$. This encoding performed well in our experiments, but we observed slightly better performance using Sinz's sequential counter encoding [103].

To encode $\sum_{i=1}^{n} x_i \leq 1$ in the sequential counter encoding, first the new variables $s_1, \ldots, s_n$ are introduced (s_i representing that at least one of $x_1, \ldots, x_i$ are true) using the $2n - 1$ clauses

$$x_i \to s_i \quad \text{and} \quad s_{i-1} \to s_i \tag{4.3.1}$$

for $1 \leq i \leq n$ (when $i = 1$ the clause $s_0 \to s_1$ is skipped). Once the s_i variables have been introduced, $\sum_{i=1}^{n} x_i \leq 1$ is encoded using the $n - 1$ additional clauses $\neg x_i \vee \neg s_{i-1}$

for $2 \leq i \leq n$, the idea being that x_i and s_{i-1} can never both be true, because that would imply at least two variables in $x_1, \ldots, x_n$ are true. Moreover, $\sum_{i=1}^{n} x_i \geq 1$ can be encoded by setting s_n to true and adding the clauses $s_i \to (s_{i-1} \vee x_i)$ for $1 \leq i \leq n$ (when $i = 1$ the literal s_0 is left out). Setting s_n to true causes two clauses to be trivially satisfied, so they can be removed. Altogether, we encode $\sum_{i=1}^{n} x_i = 1$ using $4n - 4$ clauses.

4.3.2 Orthogonality encoding

The orthogonality of two Latin squares A and B of order n can be specified by the logical constraints

$$(A_{ij} = k \wedge A_{i'j'} = k \wedge B_{ij} = l \wedge B_{i'j'} = l) \to (i = i') \tag{4.3.2}$$

for $0 \leq i, j, i', j', k, l < n$ (cf. Zhang [123, Lemma 1]). Taking the contrapositive and writing this using the variables A_{ijk} and B_{ijk}, this is equivalent to the clauses $\neg A_{ijk} \vee \neg A_{i'j'k} \vee \neg B_{ijl} \vee \neg B_{i'j'l}$ where $0 \leq i, j, i', j', k, l < n$ with $i \neq i'$. This encoding of orthogonality uses $O(n^6)$ clauses of length 4. An alternative encoding of orthogonality that performs better in practice [123, Lemma 2] uses $O(n^4)$ clauses of length 3 and n^3 new auxiliary variables. To describe this orthogonality encoding, we follow the derivation of Bright, Keita, and Stevens [34] based on a composition square.

Consider the rows of a Latin square X of order n as a collection of n permutations of the symbols $\{0, \ldots, n-1\}$. The row inverse square X^{-1} is defined to be the Latin square whose rows are formed by the inverses of the rows of X, and the composition square XY is defined to be the square whose ith row is the ith row of X composed with the ith row of Y (in a right-to-left way). Note that the square XY is *not* a Latin square in general. We now provide a theorem that the orthogonality encoding we use relies on.

Theorem 4.3.1 (Mann [80])**.** *Two Latin squares A and B are orthogonal if and only if AB^{-1} is a Latin square.*

Let Z denote the composition square AB^{-1}, and let the Boolean variables Z_{ijk} be true exactly when the (i, j) entry of Z contains the symbol k (where $0 \leq i, j, k < n$). The square Z can be specified to be a Latin square using the encoding from Section 4.3.1. In order to encode $Z = AB^{-1}$, we need to enforce that the (i, B_{ij})th entry of Z contains the symbol A_{ij}. This is done using the clauses

$$(A_{ijk} \wedge B_{ijl}) \rightarrow Z_{ilk} \tag{4.3.3}$$

for $0 \leq i, j, k, l < n$. In fact, from $A = ZB$ we also derive the similar clause $(Z_{ilk} \wedge B_{ijl}) \rightarrow A_{ijk}$, and from $B = Z^{-1}A$ we derive $(Z_{ilk} \wedge A_{ijk}) \rightarrow B_{ijl}$. These last two types of clauses are logically redundant, but in practice they improve the performance of the SAT solver as they allow the solver to make additional useful propagations.

4.3.3 Relation encoding

Let R_1 denote the indices of the lines in the first relation, and let R_2 denote the indices of the lines in the second relation. Delisle [46] defines an equivalence class on $(\text{row}, \text{column})$ Latin square index pairs using a labelling function called RC_CLASS that we describe below. In the following, i represents a Latin square row index, and j represents a Latin square column index, so $0 \leq i, j \leq 9$. Recall that we order the lines of a 4-net(10) so that lines 0 to 9 correspond to row indices of Latin squares while lines 10 to 19 correspond to column indices of Latin squares. Under such an ordering, note that line $j + 10$ corresponds to the jth column of the Latin squares. In what follows the notation $x \leftrightarrow y$ means x and y have the same truth value (both true or both false), while $x \not\leftrightarrow y$ means x and y take opposite truth values. Delisle's

RC_CLASS(i, j) labelling function is now defined by

$$\begin{cases} 0 & \text{if } (i \in R_1 \leftrightarrow j+10 \in R_1) \text{ and } (i \in R_2 \leftrightarrow j+10 \in R_2), \\ 1 & \text{if } (i \in R_1 \leftrightarrow j+10 \in R_1) \text{ and } (i \in R_2 \not\leftrightarrow j+10 \in R_2), \\ 2 & \text{if } (i \in R_1 \not\leftrightarrow j+10 \in R_1) \text{ and } (i \in R_2 \leftrightarrow j+10 \in R_2), \\ 3 & \text{if } (i \in R_1 \not\leftrightarrow j+10 \in R_1) \text{ and } (i \in R_2 \not\leftrightarrow j+10 \in R_2). \end{cases} \tag{4.3.4}$$

Similarly, Delisle defines a labelling function ST_CLASS on symbol pairs (s, t) where s is a symbol of the first Latin square A and t is a symbol of the second Latin square B. Representing s and t as integers in $\{0, \ldots, 9\}$, note that line $s + 20$ of the net corresponds to symbol s in the first Latin square, and line $t + 30$ of the net corresponds to symbol t in the second Latin square. Concretely, ST_CLASS(s, t) is defined to be

$$\begin{cases} 0 & \text{if } (s+20 \in R_1 \leftrightarrow t+30 \in R_1) \text{ and } (s+20 \in R_2 \leftrightarrow t+30 \in R_2), \\ 1 & \text{if } (s+20 \in R_1 \leftrightarrow t+30 \in R_1) \text{ and } (s+20 \in R_2 \not\leftrightarrow t+30 \in R_2), \\ 2 & \text{if } (s+20 \in R_1 \not\leftrightarrow t+30 \in R_1) \text{ and } (s+20 \in R_2 \leftrightarrow t+30 \in R_2), \\ 3 & \text{if } (s+20 \in R_1 \not\leftrightarrow t+30 \in R_1) \text{ and } (s+20 \in R_2 \not\leftrightarrow t+30 \in R_2). \end{cases} \tag{4.3.5}$$

Delisle then notes that the condition that relations R_1 and R_2 exist in the 4-net(10) corresponding to the orthogonal Latin pair (A, B) is equivalent to the condition

$$\text{RC\_CLASS}(i, j) = \text{ST\_CLASS}(A_{ij}, B_{ij}) \text{ for all } 0 \leq i, j \leq 9. \tag{4.3.6}$$

That is, if $(A_{ij}, B_{ij}) = (s, t)$ then RC_CLASS(i, j) = ST_CLASS(s, t). We encode this condition directly into our SAT encoding. Each case 1–5 is encoded separately, so the forms of R_1 and R_2 are known, meaning that the values of RC_CLASS(i, j) and ST_CLASS(s, t) can be determined in advance for all $0 \leq i, j, s, t \leq 9$. We encode the relation constraint in contrapositive form: for all i, j, s, t with ST_CLASS$(s, t) \neq$ RC_CLASS(i, j) we enforce that $(A_{ij}, B_{ij}) \neq (s, t)$, i.e., $A_{ij} \neq s$ or $B_{ij} \neq t$. In

Boolean logic, this becomes the clauses

$$\neg A_{ijs} \vee \neg B_{ijt} \text{ where } \mathrm{RC\_CLASS}(i,j) \neq \mathrm{ST\_CLASS}(s,t) \tag{4.3.7}$$

for all $0 \leq i, j, s, t \leq 9$.

4.3.4 Symmetry breaking

In this section we describe how we encode Delisle's symmetry breaking propositions into conjunctive normal form. In particular, we encode properties of minimal pairs of orthogonal Latin squares with two relations described in Section 4.2.1, as such constraints do not remove any solutions up to isomorphism. Thus, in this section we assume that (A, B) is a minimal pair of orthogonal Latin squares.

First, consider Proposition 4.2.3, which applies in all cases except case 4. It says that $(A_{1,0}, B_{1,0}) < (A_{0,1}, B_{0,1})$. First, we encode the weaker constraint that $A_{1,0} \leq A_{0,1}$ in conjunctive normal form via

$$\bigwedge_{\substack{0 \leq k,l \leq 9 \\ k > l}} (A_{1,0,k} \to \neg A_{0,1,l}). \tag{4.3.8}$$

Next, we encode that when $A_{1,0} = A_{0,1}$ we have $B_{1,0} < B_{0,1}$. This is done via

$$\bigwedge_{\substack{0 \leq k,l,m \leq 9 \\ k \geq l}} \big((A_{1,0,m} \wedge A_{0,1,m} \wedge B_{1,0,k}) \to \neg B_{0,1,l}\big). \tag{4.3.9}$$

Now consider Proposition 4.2.5, which says that the symbols in the first column of A appear in sorted order within the rows in the same row equivalence class. Let R denote a row equivalence class, and let $R' \coloneqq R \setminus \{\max(R)\}$. For example, in cases 1–4, the possible nonempty values for R' are $\{1,2\}$, $\{4,5\}$, and $\{7,8\}$. In case 5, the possible values for R' are $\{0\}$, $\{2\}$, $\{4\}$, and $\{6,7,8\}$. In order to ensure that the symbols in the first column of A with rows in R appear in sorted order we use the

constraints

$$\bigwedge_{\substack{i\in R'\\ 0\le l<k\le 9}} (A_{i,0,k} \to \neg A_{i+1,0,l}). \tag{4.3.10}$$

Proposition 4.2.6 says that the symbols in the first row of A appear in sorted order within the columns in the same column equivalence class and can be handled similarly. Let C denote a column equivalence class and let $C' \coloneqq C \setminus \{\max(C)\}$. We ensure the symbols in the first row of A with columns in C appear in sorted order using the constraints

$$\bigwedge_{\substack{j\in C'\\ 0\le l<k\le 9}} (A_{0,j,k} \to \neg A_{0,j+1,l}). \tag{4.3.11}$$

(In case 4, we skip the clauses from this constraint with $C' = \{0\}$, since Proposition 4.2.6 does not apply to column 0 in case 4.)

Proposition 4.2.7 says that the symbols in the same A-symbol equivalence class are sorted in the first column of A. Let S denote an A-symbol equivalence class and let $S' \coloneqq S \setminus \{\max(S)\}$. We ensure the symbols of S appear in sorted order in the first column of A using the constraints

$$\bigwedge_{\substack{s\in S'\\ 0\le i'<i\le 9}} (A_{i,0,s} \to \neg A_{i',0,s+1}). \tag{4.3.12}$$

Proposition 4.2.8 is handled in the same way. Letting T denote a B-symbol equivalence class and $T' \coloneqq T \setminus \{\max(T)\}$, we use the constraints

$$\bigwedge_{\substack{t\in T'\\ 0\le i'<i\le 9}} (B_{i,0,t} \to \neg B_{i',0,t+1}). \tag{4.3.13}$$

Finally, we discuss Proposition 4.2.4, which applies in all cases except case 2 and says that $A < B$. Since we are not in case 2, we have $A_{0,0} = B_{0,0}$ by Proposition 4.2.9. As a result, we start by considering the $(1,0)$th entries of A and B, noting that $A < B$ implies that $A_{1,0} \le B_{1,0}$. We encode $A_{1,0} \le B_{1,0}$ into Boolean logic with the

constraints

$$\bigwedge_{\substack{0\leq k,l\leq 9\\ k>l}} (A_{1,0,k} \to \neg B_{1,0,l}). \tag{4.3.14}$$

Next, we consider the case when $A_{1,0} = B_{1,0}$. In this case, $A < B$ implies that $A_{2,0} \leq B_{2,0}$, and we encode $A_{1,0} = B_{1,0} \to A_{2,0} \leq B_{2,0}$ into Boolean logic with the constraints

$$\bigwedge_{\substack{0\leq k,l,m\leq 9\\ k>l}} \big((A_{1,0,m} \wedge B_{1,0,m} \wedge A_{2,0,k}) \to \neg B_{2,0,l}\big). \tag{4.3.15}$$

We could continue in this fashion and also encode constraints corresponding to $(A_{1,0} = B_{1,0} \wedge A_{2,0} = B_{2,0}) \to A_{3,0} \leq B_{3,0}$, etc. However, in practice it was sufficient to only consider the entries to the $(2, 0)$th entry. In other words, we did not encode the full constraint $A < B$ in our SAT instances but the strictly weaker constraint $[A_{0,0}, A_{1,0}, A_{2,0}] \leq [B_{0,0}, B_{1,0}, B_{2,0}]$.

4.4 Exhaustive enumeration and proof generation

This section explains the process by which we use a SAT solver to find all solutions of a SAT instance (see Section 4.4.1) and the process by which we generate and check proof certificates that the enumeration was performed correctly, with no missing solutions (see Section 4.4.2).

4.4.1 Exhaustive enumeration

Typical modern SAT solvers stop as soon as a satisfying assignment is found and do not support exhaustively enumerating all solutions of a SAT instance. However, the IPASIR-UP interface [53], as supported by the SAT solver CaDiCaL [19], enables us to find all solutions. IPASIR-UP is an interface that can be used to inject custom code into a SAT solver in order to change its behaviour. One function supported by IPASIR-UP is `cb_check_found_model`, a function that is called when the SAT solver has found a new solution. Inside `cb_check_found_model` users are able to add code that modifies a SAT instance whenever a solution is found.

In our case, we add a "blocking clause" into the SAT instance every time a solution is found that prevents the solution from occurring again. Once the blocking clause has been added, the solver continues looking for a new solution. Eventually, once all solutions have been found, the solver reports that the updated instance (i.e., the instance augmented with all blocking clauses) is *unsatisfiable*—it has no solutions.

The blocking clause that we inject into the SAT instance must only block the single solution that was found and no others. Suppose (S, T) is the pair of orthogonal Latin squares that the solver found. We want to add the constraint

$$\neg\Big(\bigwedge_{0\le i,j\le 9}(A_{i,j,S_{i,j}} \wedge B_{i,j,T_{i,j}})\Big), \tag{4.4.1}$$

which is logically equivalent to $\bigvee_{0\le i,j\le 9}(\neg A_{i,j,S_{i,j}} \vee \neg B_{i,j,T_{i,j}})$. Note that this clause contains $2 \cdot 10^2 = 200$ literals. An observation that allows us to shorten this clause is to note that it is sufficient to block only the upper-left 9×9 entries in (S, T), because once those entries have been fixed the remaining entries are forced by the Latin square constraints. This allows us to replace the bound $0 \le i, j \le 9$ in the blocking clause with the bound $0 \le i, j \le 8$, thereby shrinking the clause to $2 \cdot 9^2 = 162$ literals.

4.4.2 Proof generation and verification

All our calls to a SAT solver will eventually finish with an unsatisfiable result (i.e., no solutions) as a result of the blocking clauses that we inject into the SAT instance in order to perform an exhaustive search. Modern SAT solvers such as CaDiCaL support generating a "proof certificate" of unsatisfiability. The proof certificate contains a log of the deductions that the solver made in order to determine that the SAT instance has no solutions. The certificate can then be checked by a proof verifier, a separate program that verifies each deduction in the proof logically follows from the previous deductions.

The certificates we generate are based on the DRAT proof format [118]. A DRAT certificate consists of the list of clauses deduced by the solver in the order in which they were deduced. The final clause in an unsatisfiability certificate will be the empty

clause. The empty clause being a logical consequence of the original SAT instance proves that the original instance was unsatisfiable, since no truth assignments satisfy the empty clause.

Every step in a DRAT proof is classified as either an *addition*—a clause that can be deduced from previously deduced clauses or clauses in the original SAT instance—or a *deletion*, a clause that was previously added but is no longer needed and should be removed. Our work uses a simple extension of the DRAT format first proposed by Bright et al. [29]. In this extension a third kind of step is supported, a *trusted addition*—a clause that will be added into the list of clauses in the proof even though it cannot necessarily be deduced from the previous clauses in the proof.

We require trusted clauses in our proofs because the blocking clauses we used to perform exhaustive enumeration were added through the IPASIR-UP interface, not deduced by the solver, and therefore cannot be derived through the typical logical deduction process. Thus, in our DRAT proofs when a blocking clause is generated a trusted addition is written into the DRAT proof.

A *proof verifier* takes as input the SAT instance and a DRAT proof of unsatisfiability and verifies every addition step in the proof logically follows from the current set of derived clauses in conjunction with the clauses in the original SAT instance. Deletion steps remove clauses in the current set of derived clauses when they are no longer needed in order to improve the efficiency of the proof verifier. Trusted addition steps add a clause into the current set of derived clauses without verifying its deducibility.

4.5 Results

We now discuss our computational results enumerating all 4-nets(10) with at least two nontrivial relations. Our results were run on an Intel i7 CPU running at 2.8 GHz and using the SAT solver CaDiCaL 1.9.4 using up to 250 MiB of memory. Our scripts are freely available and archived on Zenodo at doi.org/10.5281/zenodo.17352786.

We use a Python script to generate SAT instances in each of the five possible forms (cases 1–5) following the encoding described in Section 4.3. In order to mitigate

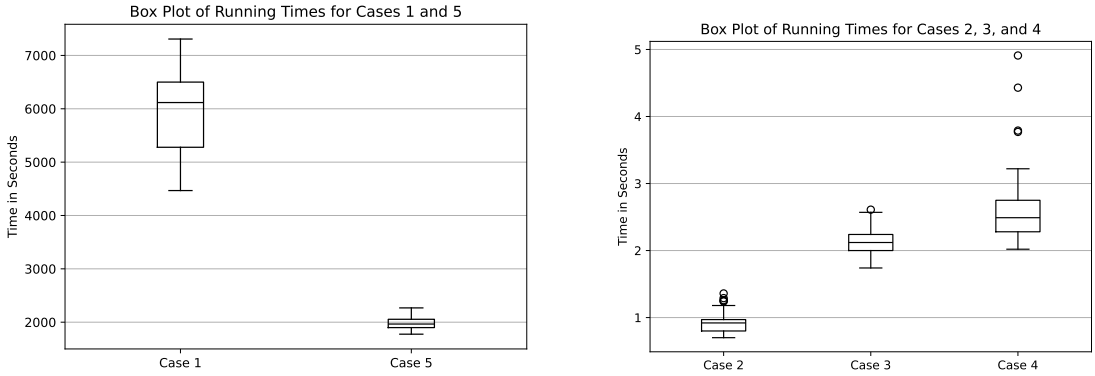
the effect of randomness in the search, each case was independently solved 45 times, each time with a different random seed. The differing random seeds ensure that each instance of CaDiCaL makes different choices during the solving process. A tabular summary of the results of these trials is provided in Table 4.5.1, and a box plot of the running times is provided in Figure 4.5.1.

The SAT solver determined that cases 2, 3, and 4 all had no solutions and these cases were always solvable in a few seconds. It is interesting to note that Delisle [46] ruled out cases 2 and 4 theoretically using a counting argument, but despite the fact we did not explicitly use this in the SAT encoding, the SAT solver quickly proved unsatisfiability on its own. The fact that the solver also quickly ruled out case 3 suggested to us that case 3 was also resolvable using a counting argument, and we were successful in finding one (included at the end of this section). Unfortunately, the proofs produced by the SAT solver, while logically correct, are not intended to be human-readable and the contents of the proofs did not provide us with any mathematical insight.

In case 1 the SAT solver found 3,904 solutions, and in case 5 the SAT solver found 22,320 solutions. The latter count agrees with the count reported by Delisle, but the former count is exactly half of Delisle's count. We contacted Delisle and Myrvold (who ran independent searches for 4-nets(10) with two relations) and their complete enumeration in cases 1 and 5 matched ours exactly, so the count reported by Delisle in case 1 was simply a misprint. We also verified that up to main class equivalence there are exactly 91 solutions (7 in case 1 and 84 in case 5) and that 6 of these (all in

Table 4.5.1: A summary of the running times (in seconds) of the 45 instances run in each of the five cases. The minimum DRAT proof size is also provided as well as the number of solutions found in each case.

case	mean	median	minimum	maximum	proof size	solutions
1	5971.2	6116.8	4467.2	7307.3	3.6 GiB	3904
2	0.9	0.9	0.7	1.4	2.1 MiB	0
3	2.1	2.1	1.7	2.6	4.1 MiB	0
4	2.7	2.5	2.0	4.9	5.3 MiB	0
5	1981.2	1965.4	1775.2	2267.8	1.6 GiB	22320



(a) Cases 1 and 5 (b) Cases 2, 3, and 4

Fig. 4.5.1: A box plot visualization of the running times of the 45 instances solved in each case.

case 5) are of rank 34 while the other 85 are of rank 35.

As mentioned in Section 4.4.2, CaDiCaL was configured to generate DRAT proofs and each time a blocking clause was generated a "trusted addition" clause was added to the proof. Since our proofs in cases 1 and 5 use trusted additions for the blocking clauses, they cannot be verified using a standard DRAT proof checker like DRAT-trim [118]. However, DRAT-trim-t (a fork of DRAT-trim) supports trusted additions, so we verified all our proofs using DRAT-trim-t [27]. The proof size of the shortest proof produced in each case is given in Table 4.5.1. The proofs for cases 2–4 were all verified in under a second, while the proof in case 1 was verified in 3.2 hours and the proof in case 5 was verified in 0.9 hours.

The fact that the SAT solver was able to quickly rule out case 3 inspired us to look for a counting argument that could rule out this case. We were successful using an approach similar to the arguments used by Gill and Wanless to count the number of points of certain type in a net [58, Thm. 3.1].

Theorem 4.5.1. *There exist no 4-nets(10) with two relations in cases 2–4.*

Proof. Say R_1 and R_2 are two relations in a 4-net(10). In what follows we use the relational code '0' to denote $R_1 \cap R_2$, '1' to denote $R_1 \setminus R_2$, '2' to denote $R_2 \setminus R_1$, and '3' to denote $\overline{R_1} \cap \overline{R_2}$. The *type* of a point is a 4-character $\{0,1,2,3\}$-string denoting

the point's relational codes from each parallel class. For example, a point of type 1122 is in R_1 but not R_2 in the first two classes, and is in R_2 but not R_1 in the last two classes. Let t_{ijkl} denote the number of points in the net of type $ijkl$. Note that for R_2 to be a relation it must be the case that $t_{ijkl} = 0$ when $i + j + k + l \not\equiv 0 \pmod 2$, and similarly for R_1 to be a relation it must be the case that $t_{ijkl} = 0$ when $\lfloor i/2 \rfloor + \lfloor j/2 \rfloor + \lfloor k/2 \rfloor + \lfloor l/2 \rfloor \not\equiv 0 \pmod 2$, so there are $4^4/4 = 64$ nonzero t_{ijkl} variables.

Let $[[x_1, y_1, z_1], [x_2, y_2, z_2], [x_3, y_3, z_3], [x_4, y_4, z_4]]$ denote the form of R_1 and R_2 as defined in Section 4.2. Now count the number of points of type $00{*}{*}$ where the symbols '$*$' are arbitrary. Note there are exactly x_1x_2 points which lie on both a line with index in $[0, x_1)$ and a line with index in $[10, 10 + x_2)$, so

$$\sum_{0 \le k,l \le 3} t_{00kl} = x_1x_2. \tag{4.5.1}$$

Similar equations can be derived by counting points of other types. For example, there are exactly x_1x_3 points of type $0{*}0{*}$, exactly y_1y_4 points of type $1{*}{*}1$, and exactly z_1x_2 points of type $20{*}{*}$, resulting in the equations

$$\sum_{0 \le j,l \le 3} t_{0j0l} = x_1x_3, \quad \sum_{0 \le j,k \le 3} t_{1jk1} = y_1y_4, \quad \sum_{0 \le k,l \le 3} t_{20kl} = z_1x_2. \tag{4.5.2}$$

In case 3, the linear system corresponding to the $3^2\binom{4}{2} = 54$ ways of fixing two entries in the point type to values in $\{0, 1, 2\}$ when converted into reduced row echelon form has a row corresponding to $t_{0123} - t_{3210} = -1/2$ which has no integer solutions.

In cases 2 and 4, the linear systems corresponding to the $4^2\binom{4}{2} = 96$ ways of fixing two entries in the point type to values in $\{0, 1, 2, 3\}$ are both inconsistent over the reals. □

The counting argument in the proof of Theorem 4.5.1 provides some theoretical conditions speculated on by Delisle in their original work:

Interestingly, no pairs of MOLS are completable for case 3. Some theoretical

conditions possibly exist to explain this, but are not known at this time. [46]

4.6 Conclusion

In this chapter we recreated Delisle's 2010 enumeration of all 4-nets(10) with two nontrivial relations. In contrast to Delisle's original search that used a custom-written backtracking program, we use a SAT solver and found that the SAT solver could complete the search over 6000 times faster (when run on modern hardware) than the original backtracking code. This is in part due to improvements in processing power, but it is also due to the powerful search-with-learning algorithms used in modern SAT solvers that can be effective at solving problems in design theory. For example, the author of the SAT solver SATO, H. Zhang, observed SAT solvers are particularly effective at solving Latin square problems:

> *In the earlier stage of our study of Latin square problems, the author wrote two special-purpose programs. After observing that these two programs could not do better than SATO, the author has not written any special-purpose search programs since then.* [124]

Moreover, our results are more trustworthy in the sense that they do not require trusting the implementation of a search algorithm. Instead, we generate certificates that our search was exhaustive, and our results only require trusting the reduction of the problem into Boolean logic (as described in Section 4.3) and the proof verifier that we use (as described in Section 4.4.2).

For future work, it would be interesting to use a SAT solver to investigate the results of Gill and Wanless [58] who enumerated all 4-nets(10) with a single nontrivial relation and ruled out the existence of a relation of type $[2, 2, 2, 4, 6]$ in a 5-net(10). More potential future work would be to rule out the existence of other relation types in a 5-net(10) or a 6-net(10), or to rule out the existence of two nontrivial relations in a 5-net(10) or a 6-net(10). If the latter was accomplished, the work of Howard [60] would imply that 4 MOLS(10) do not exist. SAT solvers might also be useful in

exploring the existence of other combinatorial designs whose existence is uncertain, such as non-Desarguesian projective planes of order 11.

CHAPTER 5

Transitive sets of MOLS and MacNeish's conjecture

We investigate MacNeish's conjecture in a symmetry-restricted setting by requiring that the autotopy group of a set of MOLS acts transitively on the corresponding orthogonal array. This transitive condition is natural from the viewpoint of orthogonal arrays and isotopy, and it isolates highly structured sets of MOLS. Our central theme is that transitivity forces sets of MOLS to carry rigid group-theoretic data.

We exploit a precise equivalence between transitive sets of MOLS and a purely group-theoretic object we call a group packet. This correspondence provides a unified framework for proving reduction theorems towards MacNeish's conjecture in the transitive setting and implementing a systematic computational search for transitive sets of MOLS by enumerating subgroup configurations in finite groups. In the simply transitive case, this framework becomes especially effective: it reduces the existence problem to searching among groups of order n^2, and it allows us to prove that MacNeish's conjecture holds for simply transitive sets of MOLS.

Our contributions are as follows.

1. We introduce the notions of transitive and simply transitive sets of MOLS via the action of the autotopy group on the orthogonal array, and formalize its properties within the orthogonal array framework.

2. We define group packets (and their nontrivial equivalence) and prove a classification theorem: isotopy classes of transitive elements of sets of q MOLS of order

n are in bijection with equivalence classes of group packets.

3. We further restrict to the simply transitive case, identifying simply transitive sets of MOLS with disjoint group packets.

4. We apply the group packet framework to MacNeish's conjecture in the transitive setting. We prove reduction theorems that bound q under natural hypotheses on the Sylow structure of the ambient group packet, and in particular, we prove that MacNeish's conjecture holds for simply transitive sets of MOLS.

5. We develop and implement a computational pipeline (SageMath with GAP) that enumerates subgroup configurations inside groups of order kn^2, constructs the induced Latin squares, tests associativity in a group theoretic sense, and classifies non-associative squares into main classes via canonical graph certificates.

This chapter reproduces *Transitive Sets of Mutually Orthogonal Latin Squares*, co-authored with Ilya Shapiro, and submitted to a journal (also available on arXiv [69]). It is included verbatim, with some formatting changes to ensure consistency, as part of this dissertation.

5.1 Introduction

Write the distinct prime-power factorization of n as $p_1^{v_1}p_2^{v_2}\cdots p_u^{v_u}$ and define

$$f(n) = \min\{p_1^{v_1}, p_2^{v_2}, \ldots, p_u^{v_u}\} - 1.$$

Classical works of MacNeish [79] and Mann [80] guarantee at least $f(n)$ MOLS of order n. It was natural to wonder whether this lower bound was tight for all n. If that were the case, Euler's conjecture (as cited in [92]) would follow: there would be no orthogonal pair whenever $n \equiv 2 \pmod 4$, since then $f(n) = 1$. Let $N(n)$ denote the maximum number of MOLS of order n. MacNeish's conjecture states that $N(n) = f(n)$ but Parker [94] demonstrated that in some instances $N(n) > f(n)$, disproving the hoped-for equality in general.

Bose and Shrikhande [25] generalized Parker's ideas and produced infinite families of counterexamples to Euler's conjecture, including all n of the form $36w + 22$, w a nonnegative integer. They also compiled for $n \leq 150$ a list of orders with $N(n) > f(n)$ and gave an explicit orthogonal pair for $n = 22$, the smallest counterexample they established. More broadly, Bose and Shrikhande [25] provided improved lower bounds for $N(n)$ for many $n \geq 22$.

The MacNeish–Mann bound states $N(n) \geq f(n)$. Bose and Shrikhande [25] further obtained the following product inequality: If $n = n_1 n_2 \cdots n_u$, then

$$N(n) \;\geq\; \min\{N(n_1), N(n_2), \ldots, N(n_u)\}.$$

Their argument uses a general result on orthogonal arrays due to Bose [22] and Bush [39]. Let $A = \{a_{ij}\}$ be a matrix with entries in $\{0, 1, \ldots, n-1\}$ having r rows and $m = \lambda n^d$ columns. For any choice of $d \leq r$ rows, view each column of the resulting $d \times 1$ submatrix as an ordered d-tuple. We call A an orthogonal array of index λ if every d-tuple over the set $\{0, \ldots, n-1\}$ appears exactly λ times among these columns.

It is well-known [40]; [68, Thm. 11.1.1] that an orthogonal array of $q+2$ constraints, n levels, strength 2, and index 1 exists if and only if there is a set of q MOLS of order n. Since $N(21) \geq 4$ [94], a formulation of the equivalence introduced in [22, 39] can be used to obtain

$$N(105) \;\geq\; \min\{N(21), N(5)\} \;\geq\; 4,$$

whereas

$$f(105) = \min\{3, 5, 7\} - 1 = 2.$$

Using the method of differences, Parker [95] also proved $N(n) \geq 2$ for $n = \frac{1}{2}(3q-1)$ with q a prime power satisfying $q \equiv 3 \pmod 4$—in particular for $n = 10$. Bose, Shrikhande, and Parker [26] strengthened the main theorem of [25], obtained sharper bounds for $N(n)$, established $N(n) \geq 2$ for $n \in \{14, 26\}$ and for $n = 12t + 10$ where t is a nonnegative integer, and showed that Euler's conjecture fails for all $n = 4t + 2 > 6$.

Euler's conjecture—that a Latin square of order $4n + 2$ has no orthogonal mate—

was refuted for every $n \geq 2$ by Bose, Shrikhande, and Parker [26]. Their work was inspired by Parker's [94] 1959 construction of four MOLS of order 21, which provided the first counterexample to MacNeish's conjecture. Since MacNeish's conjecture generalizes Euler's, and Parker's example disproving MacNeish's conjecture used the group $\mathbb{Z}_{21}$, it is natural to seek counterexamples from Latin squares based on groups to Euler's conjecture. In other words, are there admissible groups of order n when $n \equiv 2 \pmod 4$?

Interestingly, Fleisher's 1934 dissertation [55] and Mann's [80] 1942 construction established that Latin squares based on groups satisfy Euler's conjecture. Mann's proof was later rediscovered by Jungnickel [66]. In a subsequent work, Mann [82] identified structural conditions on a Latin square that preclude the existence of an orthogonal mate.

In this chapter we restrict sets of MOLS, requiring the autotopy group to act transitively (see Definition 5.3.1), which we propose ensures that MacNeish's conjecture holds. We do prove the conjecture for simply transitive sets (Definition 5.3.2; Theorem 5.4.6). Table 5.5.2 shows that the large sets of MOLS that we examined are not transitive.

The chapter is organized as follows. Section 5.2 reviews the correspondence between sets of MOLS and orthogonal arrays and also introduces autotopy groups (see Definition 5.2.4) in this context. In Section 5.3 we define transitive (Definition 5.3.1) and simply transitive (Definition 5.3.2) sets of MOLS, introduce group packets (Definition 5.3.4), establish a bijective correspondence between transitive MOLS and group packets (Theorem 5.3.17), and describe its restriction to the simply transitive case (Corollary 5.3.18). Section 5.4 applies this restricted correspondence to MacNeish's conjecture: general reduction results are obtained (Theorems 5.4.4 and 5.4.7) and a proof that the conjecture holds for simply transitive sets of MOLS (Theorem 5.4.6) is given. Section 5.5 collects structural results on autotopy groups of Latin squares based on groups and sets of MOLS in which each Latin square is based on a group, and describes computational methods for constructing and classifying Latin squares arising from group packets. Table 5.5.1 summarizes the classifications of Latin squares

of orders at most 11 according to the action of their autotopy groups. For each order, we list the number of main classes and then record how many of these are based on a group, how many are simply transitive but not based on a group, how many are transitive but not simply transitive, and how many are non-transitive. Table 5.5.2 summarizes examples of large sets of MOLS that have non-prime power orders together with the order of their autotopy groups. For each order, we record the size of the MOLS set, the order of the autotopy group of the MOLS set, and cite sources of the constructions.

5.2 Sets of MOLS and orthogonal arrays

We recall the definition of an orthogonal array.

Definition 5.2.1. *An orthogonal array of size D with c constraints, n levels, strength d, and index λ is a $c \times D$ matrix A having n different elements and with the property that each different ordered d-tuple of elements occurs exactly λ times as a column in any d-rowed submatrix of A [68], it follows that $D = \lambda n^d$.*

There is a well-known correspondence between sets of MOLS and orthogonal arrays. Specifically, an orthogonal array of $q+2$ constraints, n levels, strength 2, and index 1 is equivalent to a set of q MOLS of order n [40]; [68, Thm. 11.1.1]. From now on, we specialize in this case and so use the following definition.

Definition 5.2.2. *An element of $(q+2)$-OA(n) is an orthogonal array, i.e., the data of $(S, X_i, \pi_i)_{i=1}^{q+2}$ where $\pi_i \colon S \to X_i$ such that*

$$\pi_{ij} = \pi_i \times \pi_j \colon S \to X_i \times X_j$$

is a bijection for all $i \neq j$.

It is immediate that all X_i have the same size and we require $|X_i| = n$.

Definition 5.2.3. *An isotopy between orthogonal arrays (S, X_i, π_i) and (S', X_i', π_i') is*

a collection of bijections (σ, σ_i) *with* $\sigma\colon S \to S'$ *and* $\sigma_i\colon X_i \to X_i'$ *such that*

$$\pi_i'\sigma = \sigma_i\pi_i.$$

We note that our $(q+2)$-OA(n) would normally be denoted by OA$(n^2, q+2, n, 2)$.

Definition 5.2.4. *The* autotopy group *of an orthogonal array* $(S, X_i, \pi_i)_{i=1}^{q+2}$ *is*

$$\mathrm{Aut}(S, X_i) = \Big\{ \sigma = (\sigma_i) \in \prod_{i=1}^{q+2} \Sigma_{X_i} : \sigma(S) = S \Big\},$$

where Σ_{X_i} *is the group of bijections from* X_i *to* X_i *and* S *is viewed as a subset of the product via* $\prod \pi_i\colon S \to \prod X_i$*. In fact, we will sometimes denote the array by*

$$S \subset \prod_{i=1}^{q+2} X_i.$$

Note that the autotopy group is just the group of isotopies from an array to itself. The reader can observe that by definition, $\mathrm{Aut}(S, X_i)$ acts on the sets S and X_i (via σ_i on the latter). Recall that for a group G acting on sets X and Y, a map $f\colon X \to Y$ is G-equivariant if $f(gx) = gf(x)$ for all $g \in G$; the maps π_i are $\mathrm{Aut}(S, X_i)$-equivariant. Moreover, σ determines the σ_i's uniquely, and conversely, for any $i \neq j$, knowing σ_i and σ_j determines the rest of the structure of the autotopy/isotopy.

5.3 Transitive MOLS and groups

In this section, we introduce our restricted class of sets of MOLS and show that they can be studied using group theoretic data.

Definition 5.3.1. *We say that a set of MOLS is* transitive *if any element of the orthogonal array corresponding to the set of MOLS can be sent to any other using an element of the autotopy group of the set of MOLS. More precisely, we require that* $\mathrm{Aut}(S, X_i)$ *act transitively on* S*. We also say that the orthogonal array is transitive.*

We can further restrict our setting to the following:

Definition 5.3.2. *We say that a set of MOLS is* simply transitive *if its autotopy group has a subgroup such that any element of the orthogonal array corresponding to the set of MOLS can be sent to any other using precisely one element of the subgroup. More precisely, we require that there is a subgroup $G \leqslant \mathrm{Aut}(S, X_i)$ that acts simply transitively on S. We also say that the orthogonal array is simply transitive.*

A simply transitive set of MOLS is transitive.

Remark 5.3.3. *Note that both notions—transitive and simply transitive sets of MOLS—are closed under products of sets of MOLS.*

We note that there are transitive MOLS that are not simply transitive. See Table 5.5.1, where we note the existence of a Latin square of order 9 and a Latin square of order 10 which are transitive but not simply transitive. Further details regarding their construction can be found in Examples 5.5.11 and 5.5.13.

We now introduce the group theoretic data that will be shown to characterize the transitivity of MOLS.

Definition 5.3.4. *We call the data $(G, H_i)_{i=1}^{q+2}$ a* group packet*, an element of $(q+2)$-GP(n), if G is a group, $H_i \leqslant G$ are subgroups such that there is a subgroup K with*

$$H_i \cap H_j = K$$

for all $i \neq j$ and the indices are as follows:

$$[G : H_i] = [H_i : K] = n.$$

We sometimes write $(G, H_i) = (G, H_i)_{i=1}^{q+2}$ for brevity.

Definition 5.3.5. *Given two group packets (G, H_i) and (G', H_i'), an* admissible morphism

$$\alpha \colon (G, H_i) \to (G', H_i')$$

is a group homomorphism $\alpha \colon G \to G'$ such that $\alpha(H_i) \subset H_i'$ for all i; and the induced

maps

$$\bar{\alpha}\colon G/H_i \to G'/H_i'$$

$$gH_i \mapsto \alpha(g)H_i'$$

are bijections for all i.

Definition 5.3.6. *Two group packets (G, H_i) and (G', H_i') are said to be* equivalent *if there is a third group packet (G'', H_i'') and admissible morphisms α, β such that:*

$$(G, H_i) \xrightarrow{\alpha} (G'', H_i'') \xleftarrow{\beta} (G', H_i').$$

Definition 5.3.6 may cause concern as it might not be immediately obvious why what is described is an equivalence relation, i.e., why is it transitive? However, this is addressed in Corollary 5.3.16 below.

Definition 5.3.7. *We say that a group packet (G, H_i) is* disjoint *if the intersections between H_i's are trivial, i.e., $K = \{e\}$.*

Remark 5.3.8. *It is immediate that given group packets $(G^{(\alpha)}, H_i^{(\alpha)})_{i=1}^{q+2}$ we can form a group packet*

$$\left(\prod_\alpha G^{(\alpha)}, \prod_\alpha H_i^{(\alpha)}\right)_{i=1}^{q+2}$$

and that if the original packets are all disjoint, then so is the new one.

Lemma 5.3.9. *Let (S, X_i, π_i) with $|X_i| = n$ be a transitive orthogonal array. Choose $s \in S$, and set $x_i = \pi_i(s) \in X_i$. Then*

$$G = \mathrm{Aut}(S, X_i) \quad K = \mathrm{Stab}_G(s) \quad H_i = \mathrm{Stab}_G(x_i)$$

is a group packet in $(q+2)$-GP(n).

Proof. With G, H_i, K as in the statement of the Lemma, we have that $\pi_i\colon S \to X_i$ is G-equivariant and since the action of G on S is transitive, then so it is on X_i. Note

that $K \leqslant H_i$ so that for every i we have a commutative diagram:

$$\begin{array}{ccc} S & \xleftarrow[gs \longleftarrow\!\shortmid gK]{\simeq} & G/K \\ {\scriptstyle \pi_i}\downarrow & & \downarrow{\scriptstyle gK \mapsto gH_i} \\ X_i & \xleftarrow[gx_i \longleftarrow\!\shortmid gH_i]{\simeq} & G/H_i \end{array} \tag{5.3.1}$$

and we immediately get $[G : H_i] = |X_i| = n$. Since $\pi_{ij} \colon S \to X_i \times X_j$ is a G-equivariant bijection, so

$$H_i \cap H_j = \mathrm{Stab}_G(x_i, x_j) = \mathrm{Stab}_G(s) = K$$

and furthermore, $[H_i : K] = n$. Thus, we obtained a group packet (G, H_i), an element of $(q+2)$-GP(n). □

Lemma 5.3.10. *Let (G, H_i) be a group packet in $(q+2)$-GP(n), then*

$$G/K \subset \prod_{i=1}^{q+2} G/H_i$$

is a transitive orthogonal array in $(q+2)$-OA(n).

Proof. Given a group packet (G, H_i) in $(q+2)$-GP(n), let $S = G/K$ and $X_i = G/H_i$ (so that $|X_i| = n$). We have

$$G/K \subset \prod_{i=1}^{q+2} G/H_i$$

$$gK \mapsto (gH_i)_{i=1}^{q+2}$$

since $H_i \cap H_j = K$ implies that G/K embeds into $G/H_i \times G/H_j$ for any $i \neq j$. Furthermore,

$$|G/K| = [G : H_i][H_i : K] = n^2 = [G : H_i][G : H_j] = |G/H_i \times G/H_j|$$

so that the embedding is actually a bijection and (S, X_i) is an orthogonal array in $(q+2)$-OA(n).

Note that there is a group homomorphism

$$f\colon G \to \mathrm{Aut}(G/K, G/H_i)$$

$$g \mapsto f_g(yK) = gyK$$

and since G acts transitively on G/K, so does $\mathrm{Aut}(G/K, G/H_i)$, thus, we obtained a transitive orthogonal array. □

Remark 5.3.11. *If we relax the definition of a group packet $(G, H_i)_{i=1}^{q+2}$ so that $H_i \leqslant G$ with*

$$|H_i \cap H_j| \leq k$$

for all $i \neq j$ and $|G| = kn^2$, while $[G : H_i] = n$, then

$$G \to \prod_i G/H_i$$

is a transitive element of $OA(kn^2, q+2, n, 2)$, i.e., the index of the array is now k. If (G, H_i) is indeed a group packet, then the resulting array of index k can be "folded" into one of index 1.

Note that in light of the other conditions, $|H_i \cap H_j| \leq k$ is equivalent to $|H_i \cap H_j| = k$. Indeed, $G/(H_i \cap H_j)$ embeds into $G/H_i \times G/H_j$ and while the latter has size n^2, the former has size at least n^2.

Remark 5.3.12. *Note that the kernel of*

$$f\colon G \to \mathrm{Aut}(G/K, G/H_i)$$

is

$$K_0 = \bigcap_{g \in G} gKg^{-1}.$$

Furthermore, the packet $(G/K_0, H_i/K_0)$ produces a transitive orthogonal array that is canonically isotopic to the one produced by (G, H_i). Thus, if G is Abelian, then its group packet can be replaced with one that has a trivial K.

Definition 5.3.13. *We say that a group packet* (G, H_i) *is* reduced *if*

$$\bigcap_{g \in G} gKg^{-1} = \{e\}.$$

Equivalently, G *embeds into* $\mathrm{Aut}(G/K, G/H_i)$.

Observe that any group packet is equivalent to a reduced one.

Remark 5.3.14. *Following Lemma 5.3.10 by Lemma 5.3.9 we have that* (G, H_i) *produces a group packet*

$$(\mathrm{Aut}(G/K, G/H_i), \mathrm{Stab}_{\mathrm{Aut}(G/K, G/H_i)} H_i)$$

and $f\colon G \to \mathrm{Aut}(G/K, G/H_i)$ *is an admissible morphism.*

Lemma 5.3.15. *An admissible morphism* $\alpha\colon (G, H_i) \to (G', H_i')$ *between two group packets induces an isotopy of the resulting (transitive) orthogonal arrays. Furthermore, we have a commutative diagram of admissible morphisms:*

$$\begin{array}{ccc} G & \xrightarrow{\quad\alpha\quad} & G' \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f} \\ \mathrm{Aut}(G/K, G/H_i) & \xrightarrow[\alpha_*]{\simeq} & \mathrm{Aut}(G'/K', G'/H_i') \end{array} \tag{5.3.2}$$

Proof. Since $\alpha(H_i) \subset H_i'$, so $\alpha(K) = \alpha(H_i \cap H_j) \subset \alpha(H_i) \cap \alpha(H_j) \subset H_i' \cap H_j' = K'$. Thus, we have a commutative diagram:

$$\begin{array}{ccc} G/K & \xrightarrow{\quad\bar{\alpha}\quad} & G'/K' \\ \downarrow{\scriptstyle \simeq} & & \downarrow{\scriptstyle \simeq} \\ G/H_i \times G/H_j & \xrightarrow[\simeq]{\bar{\alpha}\times\bar{\alpha}} & G'/H_i' \times G'/H_j' \end{array}$$

so that $\bar{\alpha}\colon G/K \simeq G'/K'$ is an isotopy.

It remains to check that for every $g \in G$ and $x'K' \in G'/K'$ we have $\alpha_*(f_g)(x'K') = f_{\alpha(g)}(x'K')$. Since $\bar{\alpha}\colon G/K \simeq G'/K'$, so we may assume that $x' = \alpha(x)$ for some

$x \in G$. We then have

$$\alpha_*(f_g)(x'K') = \alpha(gx)K' = \alpha(g)x'K' = f_{\alpha(g)}(x'K'). \qquad \square$$

Corollary 5.3.16. *The equivalence of group packets is an equivalence relation.*

Proof. The relation is clearly reflexive and symmetric, it remains to show that it is transitive. But Lemma 5.3.15, more precisely, diagram (5.3.2) shows that if two group packets are equivalent then their associated autotopy group packets are isomorphic. The converse of that statement is clear by definition. The characterization of equivalence is immediately transitive. $\square$

We are ready for the following:

Theorem 5.3.17. *There is a bijection between the set of isotopy classes of transitive elements of* q-MOLS(n) *and the set of equivalence classes of elements of* $(q+2)$-GP(n).

Proof. The constructions that go between transitive orthogonal arrays and group packets are contained in Lemmas 5.3.9, 5.3.10.

It is immediate that starting with a transitive orthogonal array, obtaining a group packet from it, and getting an orthogonal array from this group packet, we obtain an isotopic orthogonal array. This is simply diagram (5.3.1).

In the other direction, if we start with a group packet (G, H_i), obtain an orthogonal array from it, and convert the array to a group packet

$$(\mathrm{Aut}(G/K, G/H_i), \mathrm{Stab}_{\mathrm{Aut}(G/K, G/H_i)} H_i),$$

then while the packet is not the same, there is an admissible morphism f from the former to the latter, see Remark 5.3.14. Thus, the two group packets are equivalent. $\square$

We have a restricted version of the correspondence:

Corollary 5.3.18. *There is a bijection between the set of isotopy classes of simply transitive elements of* q-MOLS(n) *and the set of equivalence classes of disjoint elements of* $(q+2)$-GP(n).

Proof. Given a disjoint group packet (G, H_i), we note that it is reduced, and obtain from it an array $G \subset \prod G/H_i$. Note that G embeds into $\mathrm{Aut}(G, G/H_i)$ and obviously acts simply transitively on G. Thus, an array produced from a disjoint group packet is simply transitive.

If we have a simply transitive array, i.e., there exists a $G \subset \mathrm{Aut}(S, X_i)$ such that G acts simply transitively on S; choose $s \in S$ and set $x_i = \pi_i(s)$. The surjections π_i are $\mathrm{Aut}(S, X_i)$, thus, G-equivariant, so that if we let $H_i = \mathrm{Stab}_G(x_i)$, we obtain (as in the proof of Lemma 5.3.9, but with $K = \mathrm{Stab}_G(s) = \{e\}$) a disjoint group packet (G, H_i) whose associated orthogonal array is isotopic to (S, X_i). Recall that the isotopy is given by $g \mapsto gs$ and $gH_i \mapsto gx_i$. □

Remark 5.3.19. *Observe that both Theorem 5.3.17 and Corollary 5.3.18 respect the product of sets of MOLS on one side and the product of group packets introduced in Remark 5.3.8 on the other side.*

In light of Theorem 5.3.17, to study transitive MOLS is to study group packets. Due to the lack of control over the size of K, it is hard to bound the search, even when focusing on a specific n. Whereas by Corollary 5.3.18, to find all simply transitive MOLS of a fixed size n, one needs only to run a computer search through all the isomorphism classes of groups of size n^2. Note that there is no reason to believe that non-isomorphic groups would produce non-isotopic MOLS, but this procedure will produce them all.

To produce a transitive element of q-MOLS(n) we need to fix a k and search for group packets $(G, H_i)_{i=1}^{q+2}$ with $|G| = kn^2$; naturally, $|K| = k$.

More precisely, once such a group packet (G, H_i) is found the orthogonal array is

$$G/K \subset \prod_{i=1}^{q+2} G/H_i.$$

Remark 5.3.20. *To produce a set of MOLS some choices are required:*

We choose enumerations of G/H_i, i.e., bijections

$$\phi_i \colon \{0, 1, \ldots, n-1\} \to G/H_i$$

for every i. Then for every $3 \leq \alpha \leq q+2$ we have a Latin square

$$L_{ij}^{\alpha} := \phi_{\alpha}^{-1} \pi_{\alpha} \pi_{12}^{-1}(\phi_1(i), \phi_2(j)).$$

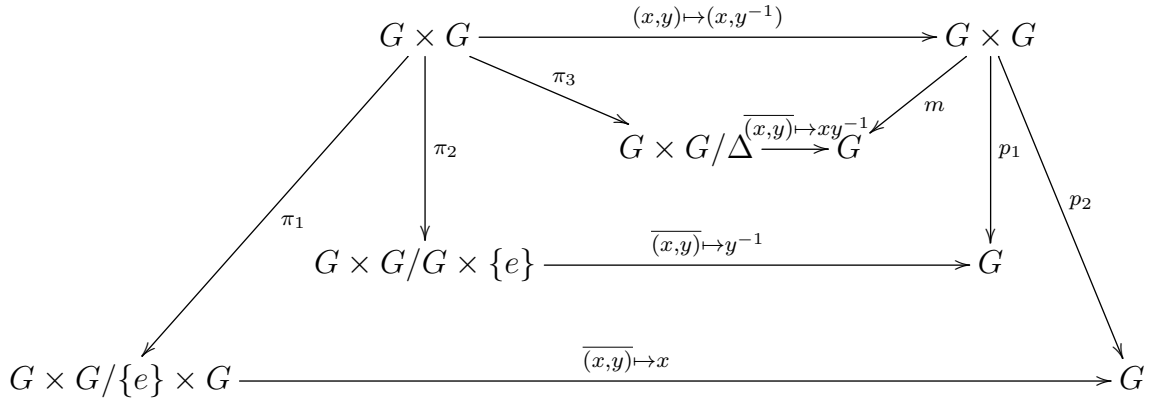
Note that the choice of ϕ_i's does not change the isotopy class of the Latin squares that appear in the set; the choice of ordering of the subgroups H_i, more precisely, the choice of which two go first, does.

5.3.1 Some simply transitive examples

A Latin square of size n is equivalent to an orthogonal array in 3-OA(n). Observe that a Latin square based on a group is isotopic to that obtained from a Cayley table of a group G. The latter is isotopic to the array obtained from the disjoint group packet

$$(G \times G, \{e\} \times G, G \times \{e\}, \Delta G)$$

where ΔG is the diagonal subgroup. More precisely, we have a commutative diagram with horizontal maps bijections:



Thus, we have shown that:

Lemma 5.3.21. *Any Latin square based on a group is simply transitive.*

Furthermore, a maximal size set of $MOLS$ for a prime power $q = p^a$, i.e., an element in $(q-1)$-MOLS(q) is obtained from the group packet

$$(\mathbb{F}_q \oplus \mathbb{F}_q, \ell_i)_{i=1}^{q+1} \tag{5.3.3}$$

where $\mathbb{F}_q$ is the finite field with q elements and ℓ_i are the 1-dimensional subspaces of the 2-dimensional $\mathbb{F}_q$-vector space. Thus, these too are simply transitive.

Remark 5.3.22. *Note that the autotopy groups for the two examples above are* $(G \times G) \rtimes \mathrm{Aut}(G)$ *and* $(\mathbb{F}_q \oplus \mathbb{F}_q) \rtimes \mathbb{F}_q^\times$ *respectively, see Theorem 5.5.1 and Corollary 5.5.6 below. Thus, in both cases the autotopy group of the array is larger than what is needed to produce the array.*

It is possible to produce a simply transitive Latin square, that does not arise as a Cayley table, see Table 5.5.1 and Examples 5.5.9 and 5.5.13.

5.4 MacNeish's conjecture

Recall that $N(n)$ is the maximum size of a set of $n \times n$ MOLS.

Conjecture 5.4.1 (MacNeish's conjecture [79])**.** *Let* $n = \prod p_j^{a_j}$ *be a distinct prime factorization of* n *and set* $p^a = \min\{p_i^{a_i}\}$*. Then* $N(n) = p^a - 1$*.*

As mentioned earlier, Parker's [94] 1959 construction of four MOLS of order 21 proves that MacNeish's conjecture fails even for group based MOLS as there is a set of 4 MOLS of order 21 based on the additive group $\mathbb{Z}_{21}$. We investigate the conjecture in the setting of transitive sets of MOLS that are not necessarily based on a group.

Remark 5.4.2. *In order to prove MacNeish's conjecture, it is sufficient to prove the following: if* q-MOLS(n) *is not empty, then for any prime* p *such that* $p^a \mid n$ *but* $p^{a+1} \nmid n$ *we have*

$$q \leq p^a - 1.$$

This follows from the product construction on sets of MOLS and the construction with (5.3.3)*, which demonstrates that* $N(n) \geq p^a - 1$*, if* $p^a = \min\{p_i^{a_i}\}$.

We need the following preparatory Lemma:

Lemma 5.4.3. *Let* (G, H_i) *be a group packet then for any* $i \neq j$ *and* $x, y \in G$ *there is a* $g \in G$ *such that*

$$xH_ix^{-1} \cap yH_jy^{-1} = gKg^{-1}.$$

Proof. Recall that the map $G/K \to G/H_i \times G/H_j$ that sends gK to (gH_i, gH_j) is a bijection. Thus, there is a $g \in G$ such that $(gH_i, gH_j) = (xH_i, yH_j)$; the result follows since

$$\mathrm{Stab}_G(gH_i, gH_j) = gKg^{-1},$$

while

$$\mathrm{Stab}_G(xH_i, yH_j) = xH_ix^{-1} \cap yH_jy^{-1}. \qquad \square$$

Both theorems below are reductions (via Sylow theorems—see Sylow's original paper [105] and modern expositions [48, 99]) to a known case (which is, anyhow, easy to show in our setting directly, see Remark 5.4.5).

Theorem 5.4.4. *Consider a transitive element of* q-MOLS(n) *and let* $(G, H_i)_{i=1}^{q+2}$ *be a group packet in the equivalence class that corresponds to this element. (Note that* G *need not be the autotopy group of the element.) For a prime* p *suppose that* $p^a \mid n$ *and* $p^{a+1} \nmid n$*, if*

$$p \nmid |K|,$$

then $q \leq p^a - 1$.

Proof. Since $p^a \mid n$ and $p^{a+1} \nmid n$, while $p \nmid |K|$, so G has a Sylow p-subgroup P with $|P| = p^{2a}$. Similarly, each H_i has a Sylow p-subgroup P_i' with $|P_i'| = p^a$. Let P_i denote a conjugate of P_i' such that $P_i \subset P$. Note that $P_i \cap P_j$ is in a conjugate of K by Lemma 5.4.3. Since $p \nmid |K|$, so $P_i \cap P_j = \{e\}$, thus, $(P, P_i)_{i=1}^{q+2}$ is a group packet with index p^a (and trivial K). We know that by Theorem 5.3.17 this would produce an

element in q-MOLS(p^a), which by a well known result $N(n) \leq n-1$ [52, 98] implies that $q \leq p^a - 1$. □

Remark 5.4.5. *It is unnecessary, but more conceptual, to use $N(n) \leq n-1$ [52, 98] in the proof of Theorem 5.4.4. We can simply do a counting argument which adds up the elements in the union of P_i's whose number is bounded by those in P: $1+(q+2)(p^a-1) \leq p^{2a}$ so*

$$(q+2)(p^a-1) \leq p^{2a} - 1 = (p^a+1)(p^a-1)$$

so $q+2 \leq p^a+1$ so $q \leq p^a-1$. A similar argument, namely $p^s+(q+2)(p^{a+s}-p^s) \leq p^{2a+s}$, where $p^s = |P_K|$ (see below), can be used in the proof of Theorem 5.4.7 below.

An immediate consequence of Theorem 5.4.4 is that if $n = \prod p_j^{a_j}$ is a distinct prime factorization of n and $p^a = \min\{p_i^{a_i}\}$, then in order to find a counterexample to Conjecture 5.4.1 in the transitive setting one need not look among group packets (G, H_i) with $|G| = kn^2$ where $p \nmid k$. In particular, $k = 1$ will never produce a counterexample:

Theorem 5.4.6. *MacNeish's conjecture holds for simply transitive sets of MOLS.*

Proof. Observe that the lower bound on $N(n)$ still holds for simply transitive sets of MOLS since a product of simply transitive sets of MOLS is still simply transitive and furthermore, the construction with (5.3.3) produces a simply transitive set of MOLS.

For the upper bound on $N(n)$, we note that a simply transitive element of q-MOLS(n) corresponds via Corollary 5.3.18 to a disjoint group packet, i.e., an element of $(q+2)$-GP(n) with a trivial K. Since $p \nmid |K|$, the result follows by Theorem 5.4.4. □

The following theorem treats another special case in which Conjecture 5.4.1 holds.

Theorem 5.4.7. *Consider a transitive element of q-MOLS(n) and let $(G, H_i)_{i=1}^{q+2}$ be a group packet in the equivalence class that corresponds to this element. For a prime p suppose that $p^a \mid n$ and $p^{a+1} \nmid n$, if P, a Sylow p-subgroup of G, is unique, i.e.,*

$$P \trianglelefteq G,$$

then $q \leq p^a - 1$.

Proof. Since P is the unique Sylow p-subgroup of G, we have that for any $T \leqslant G$ the intersection $T \cap P$ is the unique Sylow p-subgroup of T. Thus, $P_i = H_i \cap P$ and $P_K = K \cap P$ results in $P_i \leqslant P$ with $P_i \cap P_j = P_K$ for all $i \neq j$ and so $(P, P_i)_{i=1}^{q+2}$ is a group packet with index p^a (and a non-trivial K, namely P_K). This is, again, impossible unless $q \leq p^a - 1$, by Theorem 5.3.17 and $N(n) \leq n - 1$ [52, 98]. □

The proof of Theorem 5.4.4 in general only produces (P, P_i) such that $|P| = p^s(p^a)^2$, $[P : P_i] = p^a$, and $|P_i \cap P_j| \leq p^s$ for $i \neq j$. Thus, see Remark 5.3.11, we obtain an element of $\mathrm{OA}(p^s(p^a)^2, q+2, p^a, 2)$, i.e., an array of index p^s. This is insufficient to produce an effective bound on q.

5.5 Latin square methods

This section collects the main tools and procedures we use to analyse Latin squares and sets of MOLS arising from our construction. We begin by describing the autotopy group in the setting, starting from the Cayley table case and extending to sets of MOLS defined by orthomorphisms. We then present representative examples that illustrate the transitivity phenomena that occur in our output. We also describe our computational pipeline in SageMath with calls to GAP for enumerating subgroup triples, constructing and normalizing Latin squares, testing associativity, and classifying non-associative squares (more precisely, non-associative loops) into main classes via canonical graph certificates. The section concludes with summary tables of autotopy group computations.

5.5.1 Autotopy groups of MOLS based on a group

Theorem 5.5.1. *Let G be a finite group and let L be its Cayley table, considered as a Latin square, i.e.,*

$$L(x, y) = xy$$

for all $x, y \in G$. Then the autotopy group $\mathrm{Aut}(L)$ of L is isomorphic to the semi-direct product

$$(G \times G) \rtimes \mathrm{Aut}(G),$$

where $\mathrm{Aut}(G)$ acts diagonally on $G \times G$.

Proof. Let (α, β, γ) be an autotopy of L. That is, α, β, γ are bijections from G to G such that

$$\gamma(xy) = \alpha(x)\beta(y) \quad \text{for all } x, y \in G. \tag{5.5.1}$$

From (5.5.1), we get

$$\alpha(x) = \gamma(x)\beta(e)^{-1} \quad \text{for all } x \in G,$$
$$\beta(y) = \alpha(e)^{-1}\gamma(y) \quad \text{for all } y \in G.$$

Define a map

$$\varphi \colon G \to G, \quad \varphi(z) = \alpha(e)^{-1}\gamma(z)\beta(e)^{-1} \quad \text{for all } z \in G.$$

Then φ is a bijection. Moreover, for all $x, y \in G$,

$$\begin{aligned}
\varphi(xy) &= \alpha(e)^{-1}\gamma(xy)\beta(e)^{-1} \\
&= \alpha(e)^{-1}\alpha(x)\beta(y)\beta(e)^{-1} \\
&= \alpha(e)^{-1}\gamma(x)\beta(e)^{-1}\alpha(e)^{-1}\gamma(y)\beta(e)^{-1} \\
&= \varphi(x)\varphi(y).
\end{aligned}$$

Hence, $\varphi \in \mathrm{Aut}(G)$. Let $a = \alpha(e)$ and $b = \beta(e)^{-1}$. Then

$$(\alpha, \beta, \gamma) = (a\varphi,\ \varphi b^{-1},\ a\varphi b^{-1}).$$

Conversely, for any $a, b \in G$, and $\varphi \in \mathrm{Aut}(G)$, define

$$\alpha(x) := a\varphi(x), \quad \beta(y) := \varphi(y)b^{-1}, \quad \gamma(z) := a\varphi(z)b^{-1}.$$

Then α, β, γ are bijections and satisfy

$$\alpha(x)\beta(y) = a\varphi(x)\varphi(y)b^{-1} = a\varphi(xy)b^{-1} = \gamma(xy), \quad \forall x, y \in G,$$

so $(\alpha, \beta, \gamma) \in \Gamma(L)$. Hence, every autotopy of L arises uniquely this way.

Define a map

$$\theta \colon (G \times G) \rtimes \mathrm{Aut}(G) \to \mathrm{Aut}(L)$$
$$\theta(a, b, \varphi) := (a\varphi,\ \varphi b^{-1},\ a\varphi b^{-1}).$$

The group operation in the semi-direct product is given by

$$\begin{aligned}(a, b, \varphi) \cdot (a', b', \varphi') &= ((a, b)\varphi(a', b'),\ \varphi \circ \varphi') \\ &= (a\varphi(a'),\ b\varphi(b'),\ \varphi \circ \varphi'),\end{aligned}$$

which is mapped to

$$\theta\left(a\varphi(a'),\ b\varphi(b'),\ \varphi \circ \varphi'\right) = \left(a\varphi(a')\varphi \circ \varphi',\ \varphi \circ \varphi'(b\varphi(b'))^{-1},\ a\varphi(a')\varphi \circ \varphi'(b\varphi(b'))^{-1}\right).$$

Compute

$$\begin{aligned}\theta(a, b, \varphi)\theta(a', b', \varphi') &= \left(a\varphi,\ \varphi b^{-1},\ a\varphi b^{-1}\right) \circ \left(a'\varphi',\ \varphi' b'^{-1},\ a'\varphi' b'^{-1}\right) \\ &= \left(a\varphi(a'\varphi'),\ \varphi(\varphi' b'^{-1})b^{-1},\ a\varphi(a'\varphi' b'^{-1})b^{-1}\right) \\ &= \left(a\varphi(a')\varphi(\varphi'),\ \varphi(\varphi')\varphi(b'^{-1})b^{-1},\ a\varphi(a')\varphi(\varphi')\varphi(b'^{-1})b^{-1}\right) \\ &= \left(a\varphi(a')\varphi(\varphi'),\ \varphi(\varphi')(b\varphi(b'))^{-1},\ a\varphi(a')\varphi(\varphi')(b\varphi(b'))^{-1}\right)\end{aligned}$$

to show that θ is a group homomorphism. □

Remark 5.5.2. *Theorem 5.5.1 is classical in the theory of quasigroups and Latin squares, going back at least to work of Bruck [37] and Albert [13, 14]. Kotlar [71] provides a modern formulation.*

Definition 5.5.3 (Johnson, Dulmage, and Mendelsohn [65])**.** *Let G be a group.*

Suppose that a set of MOLS corresponds to an array of the form

$$(x, y, xy, x\mu_1(y), \ldots, x\mu_t(y))_{x,y\in G} \subset G^{t+3}$$

where xy denotes the group product and $\mu_i \colon G \to G$ are bijections. Such a set of MOLS is said to be based on a group*, more particularly G-based. The μ_i are called orthomorphisms.*

Lemma 5.5.4. *Any set of MOLS based on a group is isotopic to one with*

$$\mu_i(e) = e$$

for all i.

Proof. The isotopy

$$(x, y, xy, x\mu_1(y), \ldots, x\mu_t(y))_{x,y\in G} \to (x, y, xy, x\mu_1'(y), \ldots, x\mu_t'(y))_{x,y\in G}$$

where $\mu_i'(e) = e$ is given by $\mathrm{Id} \colon G \times G \to G \times G$ and $f_i(g) = g\mu_i(e)^{-1}$ for $i = 1, \ldots, t$, while f_{-2}, f_{-1}, f_0 are Id. Thus, $\mu_i'(g) = \mu_i(g)\mu_i(e)^{-1}$. □

We will assume the statement of Lemma 5.5.4 from now on.

Theorem 5.5.5. *Let G be a finite group. Let L be the Cayley table of G and let K be the Latin square defined by*

$$K(x, y) = x\mu(y),$$

with $\mu \colon G \to G$ a bijection. Then any autotopy of (L, K) is of the form

$$(\alpha,\, \beta,\, \gamma,\, \delta) = \left(a\varphi,\, \varphi b^{-1},\, a\varphi b^{-1},\, a\varphi\mu(b^{-1})\right)$$

where $a, b \in G$, $\varphi \in \mathrm{Aut}(G)$ such that

$$\varphi(\mu(y))\mu(b^{-1}) = \mu(\varphi(y)b^{-1}) \quad \text{for all } y \in G.$$

Proof. Consider the orthogonal array whose rows are $(x,\ y,\ xy,\ x\mu(y))$ for all $x, y \in G$. Let $(\alpha,\ \beta,\ \gamma,\ \delta)$ be an autotopy of the orthogonal pair. By Theorem 5.5.1, an autotopy $(\alpha,\ \beta,\ \gamma)$ of L has the form

$$(\alpha,\ \beta,\ \gamma) = (a\varphi,\ \varphi b^{-1},\ a\varphi b^{-1}),$$

for some fixed $a, b \in G$, and $\varphi \in \mathrm{Aut}(G)$.

Since (α, β, δ) is an autotopy of K, we must have

$$\delta(x\mu(y)) = \alpha(x)\mu(\beta(y)) = a\varphi(x)\mu(\varphi(y)b^{-1}) \quad \text{for all } x, y \in G. \tag{5.5.2}$$

Observe that (set $y = e$):

$$\delta(x) = \delta(x\mu(e)) = \alpha(x)\mu(\beta(e)) = a\varphi(x)\mu(b^{-1}) \tag{5.5.3}$$

so that $(x = x\mu(y))$:

$$\delta(x\mu(y)) = a\varphi(x\mu(y))\mu(b^{-1}) \tag{5.5.4}$$

Comparing (5.5.2) and (5.5.4) yields

$$a\varphi(x)\varphi(\mu(y))\mu(b^{-1}) = a\varphi(x\mu(y))\mu(b^{-1}) = a\varphi(x)\mu(\varphi(y)b^{-1}),\ \text{for all } x, y \in G,$$

which is equivalent to

$$\varphi(\mu(y))\mu(b^{-1}) = \mu(\varphi(y)b^{-1}),\ \text{for all } y \in G. \tag{5.5.5}$$

To summarize,

$$(\alpha,\ \beta,\ \gamma,\ \delta) = \left(a\varphi,\ \varphi b^{-1},\ a\varphi b^{-1},\ a\varphi\mu(b^{-1})\right)$$

and (5.5.5) is satisfied for some $a, b \in G$ and $\varphi \in \mathrm{Aut}(G)$.

On the other hand, suppose that $a, b \in G$, $\varphi \in \mathrm{Aut}(G)$ and (5.5.5) is satisfied. Let

$$(\alpha,\, \beta,\, \gamma,\, \delta) = \left(a\varphi,\, \varphi b^{-1},\, a\varphi b^{-1},\, a\varphi\mu(b^{-1})\right).$$

Theorem 5.5.1 shows that $(\alpha,\, \beta,\, \gamma)$ is an autotopy of L so it is enough to show that $(\alpha,\, \beta,\, \delta)$ is an autotopy of K. Observe that $\alpha,\, \beta,\, \delta$ are bijections and satisfy

$$\begin{aligned}\alpha(x)\mu(\beta(y)) &= a\varphi(x)\mu(\varphi(y)b^{-1}) \\ &= a\varphi(x)\varphi(\mu(y))\mu(b^{-1}) \\ &= a\varphi(x\mu(y))\mu(b^{-1}) \\ &= \delta(x\mu(y)),\end{aligned}$$

for all $x, y \in G$. □

Corollary 5.5.6. *Let G be a finite group, and let L denote its Cayley table. For $1 \le i \le t$, define Latin squares K_i by*

$$K_i(x, y) = x\mu_i(y), \qquad x, y \in G,$$

where each $\mu_i \colon G \to G$ is a bijection. Thus, $(L, K_1, \ldots, K_t)$ is based on a group. Then any autotopy of $(L, K_1, \ldots, K_t)$, is of the form

$$(\alpha,\, \beta,\, \gamma,\, \delta_1,\, \ldots,\, \delta_t) = \left(a\varphi,\, \varphi b^{-1},\, a\varphi b^{-1},\, a\varphi\mu_1(b^{-1}),\, \ldots,\, a\varphi\mu_t(b^{-1})\right),$$

where $a, b \in G$, $\varphi \in \mathrm{Aut}(G)$ such that for all i with $1 \le i \le t$,

$$\varphi(\mu_i(y))\mu_i(b^{-1}) = \mu_i(\varphi(y)b^{-1}) \quad \text{for all } y \in G.$$

Corollary 5.5.7. *Let $(L, K_1, \ldots, K_t)$ be based on a group. If μ_i are group isomorphisms, then this set of MOLS is simply transitive.*

Proof. By Corollary 5.5.6, the autotopy group contains $G \times G$. □

Corollary 5.5.8. *Let G be a finite group. The autotopy group of a G-based set of MOLS is a subgroup of $(G \times G) \rtimes \mathrm{Aut}(G)$ and contains $(G \times \{e\}) \rtimes \{e\}$.*

5.5.2 Examples of the construction

This section collects some interesting applications of our construction.

Example 5.5.9 (Simply transitive Latin square of order 6 that is not isotopic to the Cayley table of a group)**.** *Consider the symmetric group S_3 acting on $\{1,2,3\}$. Let $G = S_3 \times S_3$ where $k = 1$.*

The subgroups of G are

$$\begin{aligned} H_1 &= \langle (2,3) \rangle \times \langle (1,3,2) \rangle \cong \mathbb{Z}_6, \\ H_2 &= \langle (1,3,2) \rangle \times \langle (2,3) \rangle \cong \mathbb{Z}_6, \\ H_3 &= \langle ((1,2),(2,3)), ((1,2,3),(1,3,2)) \rangle \cong S_3. \end{aligned}$$

Furthermore, $|H_i| = 6$ and $H_i \cap H_j = \{e\} =: K$ for $i \neq j$.

We will construct a Latin square L of order 6 from the above data as described in Remark 5.3.20. Let $\mathcal{R} = G/H_1$ be the row indices of L, $\mathcal{C} = G/H_2$ be the column indices of L, and $\mathcal{S} = G/H_3$ be the symbols of L. For $g \in G/K$, consider the triple (gH_1, gH_2, gH_3). Since $H_1 \cap H_2 = \{e\}$ and $H_1 H_2 = G$, the map

$$\begin{aligned} G/K &\to \mathcal{R} \times \mathcal{C}, \\ g &\mapsto (gH_1, gH_2) \end{aligned}$$

is a bijection. Hence for each $(R, C) \in \mathcal{R} \times \mathcal{C}$ there is a unique g with $(R, C) = (gH_1, gH_2)$. We construct the Latin square L of order 6 with the symbol allocation rule

$$L(R, C) := gH_3 \in \mathcal{S}.$$

After enumerating $\mathcal{R}, \mathcal{C}, \mathcal{S}$ appropriately, the Latin square is:

$$L = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 0 & 5 & 3 & 4 \\ 2 & 0 & 1 & 4 & 5 & 3 \\ 3 & 4 & 5 & 1 & 2 & 0 \\ 4 & 5 & 3 & 0 & 1 & 2 \\ 5 & 3 & 4 & 2 & 0 & 1 \end{bmatrix}$$

Note that L is an example of a simply transitive Latin square that is not isotopic to a Cayley table of a group (as L is a non-associative loop [68]).

Remark 5.5.10. *The smallest order of a simply transitive Latin square that is not isotopic to the Cayley table of a group is* 6.

Example 5.5.11 (Transitive but not simply transitive Latin square of order 9)**.** *There is no Latin square of order* 9 *that is simply transitive but not isotopic to a Cayley table of a group. An example of a Latin square of order* 9 *that is transitive but not simply transitive is*

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 7 & 4 & 5 & 6 & 8 & 2 & 0 & 3 \\ 2 & 3 & 6 & 7 & 8 & 4 & 0 & 1 & 5 \\ 3 & 5 & 7 & 4 & 0 & 6 & 1 & 8 & 2 \\ 4 & 6 & 8 & 0 & 3 & 1 & 5 & 2 & 7 \\ 5 & 8 & 0 & 6 & 1 & 2 & 7 & 3 & 4 \\ 6 & 2 & 3 & 1 & 5 & 7 & 8 & 4 & 0 \\ 7 & 4 & 1 & 8 & 2 & 0 & 3 & 5 & 6 \\ 8 & 0 & 5 & 2 & 7 & 3 & 4 & 6 & 1 \end{bmatrix}$$

and it is constructed with $n = 9$ and $k = 3$. Its construction proceeds as in Example 5.5.9, we let

$$G = (\mathbb{Z}_3 \times (\mathbb{Z}_9 \rtimes \mathbb{Z}_3)) \rtimes \mathbb{Z}_3.$$

GAP [56] gives G in terms of its generators inside S_{18}:

$$\begin{aligned} G = \langle &(1,2,4,3,5,7,6,8,9)(10,13,17,15,18,14,12,16,11),\\ &(2,5,8)(10,11,13)(12,14,16)(15,17,18),\\ &(2,8,5)(4,7,9)(11,17,14)(13,16,18),\\ &(1,3,6)(2,5,8)(4,7,9)(10,15,12)(11,17,14)(13,18,16),\\ &(10,12,15)(11,14,17)(13,16,18)\rangle. \end{aligned}$$

The subgroups of G, written as elements inside S_{18}, are:

$$\begin{aligned} H_1 = \langle &(1,6,3)(4,9,7)(10,14,18)(11,16,15)(12,17,13),\\ &(10,15,12)(11,17,14)(13,18,16),\\ &(2,5,8)(4,9,7)(11,14,17)(13,18,16)\rangle\\ \cong &(\mathbb{Z}_3 \times \mathbb{Z}_3) \rtimes \mathbb{Z}_3\\ H_2 = \langle &(1,6,3)(4,7,9)(10,12,15)(11,17,14),\\ &(1,2,4,3,5,7,6,8,9)(10,13,17,15,18,14,12,16,11),\\ &(1,3,6)(2,5,8)(4,7,9)(10,15,12)(11,17,14)(13,18,16)\rangle\\ \cong &\mathbb{Z}_9 \rtimes \mathbb{Z}_3\\ H_3 = \langle &(1,5,4,6,2,9,3,8,7)(11,17,14),\\ &(1,6,3)(2,8,5)(4,9,7),\\ &(1,6,3)(4,7,9)(11,14,17)(13,18,16)\rangle\\ \cong &\mathbb{Z}_9 \rtimes \mathbb{Z}_3\\ K = \langle &(2,5,8)(4,9,7)(11,14,17)(13,18,16)\rangle\\ \cong &\mathbb{Z}_3. \end{aligned}$$

Remark 5.5.12. *The smallest integer n for which there exists a transitive but not simply transitive Latin square of order n is $n = 9$.*

Example 5.5.13 (Transitive and simply transitive Latin squares of order 10)**.** *Let*

$$G = D_5 \times D_5,$$

and $k = 1$*. We use the cycle notation for the elements of* D_5 *considered as a subgroup of* S_5*; the subgroups of* G *are:*

$$\begin{aligned}
H_1 &= \langle (2,5)(3,4) \rangle \times \langle (1,2,3,4,5) \rangle \cong \mathbb{Z}_{10}, \\
H_2 &= \langle (1,4,2,5,3) \rangle \times \langle (2,5)(3,4) \rangle \cong \mathbb{Z}_{10}, \\
H_3 &= \langle ((1,3)(4,5), (2,5)(3,4)), ((1,4,2,5,3), (1,2,3,4,5)) \rangle \cong D_5.
\end{aligned}$$

We proceed as in Example 5.5.9 to construct the Latin square M*, see below.*

Remark 5.5.14. *Note that both examples* $G = S_3 \times S_3 = D_3 \times D_3$ *and* $G = D_5 \times D_5$ *are equivalent to the following. Recall that* $D_n = \langle r, s \mid r^n = s^2 = (sr)^2 = 1 \rangle$ *and set* $G = D_n \times D_n$*, then* $H_1 = \langle r \rangle \times \langle s \rangle$*,* $H_2 = \langle r^2 s \rangle \times \langle r \rangle$*, and* $H_3 = \Delta D_n$ *(the diagonal embedding of* D_n *into* G*).*

Now, let

$$G' = (\mathbb{Z}_5 \times \mathbb{Z}_5) \rtimes D_4,$$

and $k = 2$*. Note that GAP [56] gives* G' *in terms of its generators inside* S_{25}*:*

$$\begin{aligned}
G' = \langle &(2,10)(3,4)(5,18)(6,11)(7,15)(9,23)(12,22)(13,16)(17,25)(19,21), \\
&(2,3)(4,6)(7,10)(8,9)(11,15)(12,14)(16,19)(17,18)(20,22)(23,24), \\
&(2,11)(3,15)(4,7)(5,25)(6,10)(8,24)(9,23)(12,22)(13,21)(14,20)(16,19)(17,18), \\
&(1,2,4,7,11)(3,5,8,12,16)(6,9,13,17,20)(10,14,18,21,23)(15,19,22,24,25), \\
&(1,3,6,10,15)(2,5,9,14,19)(4,8,13,18,22)(7,12,17,21,24)(11,16,20,23,25) \rangle \\
\cong &(\mathbb{Z}_5 \times \mathbb{Z}_5) \rtimes D_4.
\end{aligned}$$

The subgroups of G' *are (written again as elements within* S_{25}*):*

$$
\begin{aligned}
H_1' =& \langle (2,6)(3,7)(4,15)(5,17)(8,24)(10,11)(13,19)(14,20)(16,21)(18,25), \\
& (2,10)(3,4)(5,18)(6,11)(7,15)(9,23)(12,22)(13,16)(17,25)(19,21), \\
& (1,14,8,24,20)(2,18,12,25,6)(3,19,13,7,23)(4,21,16,15,9)(5,22,17,11,10)\rangle \\
\cong & D_{10}, \\
H_2' =& \langle (2,3)(4,6)(7,10)(8,9)(11,15)(12,14)(16,19)(17,18)(20,22)(23,24), \\
& (2,11)(3,15)(4,7)(5,25)(6,10)(8,24)(9,23)(12,22)(13,21)(14,20)(16,19)(17,18), \\
& (1,25,21,13,5)(2,15,23,17,8)(3,11,24,18,9)(4,19,10,20,12)(6,16,7,22,14)\rangle \\
\cong & D_{10}, \\
H_3' =& \langle (2,7,11,4)(3,6,15,10)(5,17,25,18)(8,9,24,23)(12,20,22,14)(13,19,21,16), \\
& (2,11)(3,15)(4,7)(5,25)(6,10)(8,24)(9,23)(12,22)(13,21)(14,20)(16,19)(17,18), \\
& (1,11,7,4,2)(3,16,12,8,5)(6,20,17,13,9)(10,23,21,18,14)(15,25,24,22,19)\rangle \\
\cong & \mathbb{Z}_5 \rtimes \mathbb{Z}_4, \\
K' =& \langle (2,11)(3,15)(4,7)(5,25)(6,10)(8,24)(9,23)(12,22)(13,21)(14,20)(16,19)(17,18)\rangle \\
\cong & \mathbb{Z}_2.
\end{aligned}
$$

and they are used to construct the Latin square M'. We have:

$$
M = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & 3 & 0 & 4 & 2 & 7 & 5 & 9 & 6 & 8 \\
2 & 0 & 4 & 1 & 3 & 6 & 8 & 5 & 9 & 7 \\
3 & 4 & 1 & 2 & 0 & 9 & 7 & 8 & 5 & 6 \\
4 & 2 & 3 & 0 & 1 & 8 & 9 & 6 & 7 & 5 \\
5 & 6 & 7 & 8 & 9 & 4 & 2 & 3 & 0 & 1 \\
6 & 8 & 5 & 9 & 7 & 3 & 4 & 1 & 2 & 0 \\
7 & 5 & 9 & 6 & 8 & 2 & 0 & 4 & 1 & 3 \\
8 & 9 & 6 & 7 & 5 & 1 & 3 & 0 & 4 & 2 \\
9 & 7 & 8 & 5 & 6 & 0 & 1 & 2 & 3 & 4
\end{bmatrix}
\qquad
M' = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & 0 & 6 & 9 & 8 & 7 & 3 & 4 & 5 & 2 \\
2 & 4 & 7 & 0 & 9 & 1 & 5 & 8 & 3 & 6 \\
3 & 5 & 0 & 8 & 1 & 6 & 9 & 2 & 7 & 4 \\
4 & 3 & 5 & 6 & 7 & 2 & 8 & 9 & 1 & 0 \\
5 & 2 & 9 & 4 & 3 & 8 & 0 & 1 & 6 & 7 \\
6 & 7 & 4 & 1 & 0 & 3 & 2 & 5 & 9 & 8 \\
7 & 9 & 8 & 2 & 6 & 4 & 1 & 3 & 0 & 5 \\
8 & 6 & 3 & 7 & 5 & 9 & 4 & 0 & 2 & 1 \\
9 & 8 & 1 & 5 & 2 & 0 & 7 & 6 & 4 & 3
\end{bmatrix}.
$$

Note that M is an example of a simply transitive Latin square that is not isotopic to a Cayley table of a group, and M' is an example of a transitive but not simply transitive

Latin square. We determined that they belong to different main classes by reducing them to graphs (following the method of Egan and Wanless [49]) and checking if the graphs are isomorphic using SageMath [107].

Remark 5.5.15. *The smallest order n for which there exist both a simply transitive Latin square that is not isotopic to the Cayley table of a group and a transitive but not simply transitive Latin square is $n = 10$.*

5.5.3 Computational construction and classification pipeline

We search for Latin squares that arise from triples of subgroups inside a finite group, normalize these Latin squares, and classify them. All computations were carried out in SageMath [107] with direct calls to GAP [56]. The pipeline comprises:

- enumerating candidate groups and subgroup triples;
- precomputing cosets and assembling an orthogonal-array-like incidence structure;
- converting the orthogonal-array-like incidence structure into Latin squares;
- normalizing the Latin squares;
- verifying associativity;
- grouping non-associative outputs into main classes (paratopy) via canonical graph certificates.

The program takes n and k from the user to fetch groups of order kn^2 and for each of those, we try the procedure.

5.5.3.1 Group and subgroup search

Let G be a finite group and $K \leqslant G$ a subgroup of order k. We target triples (H_1, H_2, H_3) of subgroups of order kn such that

$$H_i \cap H_j = K \qquad \text{for all } i \neq j.$$

For each order kn^2, we enumerate groups using the GAP `SmallGroup` library:

$$\texttt{NrSmallGroups}(kn^2), \qquad \texttt{SmallGroup}(kn^2, \text{index}),$$

and mapped each to a permutation group (via `IsomorphismPermGroup`) for human readability. For each G, we form

$$\mathcal{H} = \{\, H \leqslant G : |H| = kn \,\}, \qquad \mathcal{K} = \{\, K \leqslant G : |K| = k \,\}.$$

Fixing $K \in \mathcal{K}$, we filter $\mathcal{H}$ to

$$\mathcal{H}(K) = \{\, H \in \mathcal{H} : H \cap K = K \,\}.$$

We then build a graph Γ_K with vertex set $\mathcal{H}(K)$ and edges $\{H, H'\}$ whenever $H \cap H' = K$. Cliques of size 3 in Γ_K are exactly the subgroup triples (H_1, H_2, H_3) we seek. This step is implemented in Sage [107] by constructing Γ_K and extracting all 3-cliques. We seek maximum cliques when we are interested in a set of mutually orthogonal Latin squares.

To control memory, we periodically invoked GAP's [56] garbage `GASMAN('collect')`.

5.5.3.2 Construction of Latin squares

Given a fixed G and a list of candidate subgroups with their pairwise intersection K, we precompute all left cosets gH for every $g \in G/K$ and every H in the pool. We cache a dictionary $(g, H) \longmapsto gH$, with a canonical identifier for each subgroup (obtained by serializing its elements) together with the string representation of g. This precomputation eliminates repeated coset construction across different triples.

For each triple (H_1, H_2, H_3) and each $g \in G/K$, we form $\big(gH_1,\ gH_2,\ gH_3\big)$. Collecting these yields an orthogonal-array-like list of triples

$$\mathcal{T} = \{\, (R_g, C_g, S_g) : g \in G/K \,\}, \qquad R_g = gH_1,\ C_g = gH_2,\ S_g = gH_3.$$

So we convert $\mathcal{T}$ to a rectangular array by taking the distinct values of the first and second coordinates as row and column labels (sorted deterministically), and placing the third coordinate as the symbol. Concretely, in GAP [56] we set

$$\text{rows} = \{\, gH_1 : g \in G/K \,\}, \quad \text{cols} = \{\, gH_2 : g \in G/K \,\},$$

$$L(r, c) = S_g \text{ if } (R_g, C_g) = (\text{rows}[r], \text{cols}[c])$$

for a Latin square L of order n.

Entries of L are cosets. To obtain a numeric Latin square on $\{0, \ldots, n-1\}$, we reindexed symbols using the first row as reference: if the first row is $[x_0, \ldots, x_{n-1}]$, we map $x_j \mapsto j$ and apply this to all cells. We then reduce the square by permuting rows so that the first column is appropriately ordered to make L a reduced Latin square. As a safety check, we verify the Latin property (distinct symbols in every row and column) prior to reindexing.

To check whether the resulting table is isotopic to a group table, we check associativity:

$$(a * b) * c = a * (b * c) \qquad \text{for all} \quad a, b, c \in \{0, \ldots, n-1\}.$$

We also implement the classical quadrangle criterion as an optional additional filter but this is computationally expensive for large n. We record Latin squares that fail associativity as those that are not isotopic to a Cayley table of a group.

5.5.3.3 Classification of Latin squares

To identify main class representatives among non-associative outputs, we use canonical graph certificates. The reduction from a $(t-2)$-MOLS of order n to a graph is described using an orthogonal array by Egan and Wanless [49]. Let A be the orthogonal array corresponding to the $(t-2)$-MOLS of order n. Define an undirected graph G_A corresponding to A. The vertices of G_A are of three types:

- t type 1 vertices that correspond to the columns of A,
- tn type 2 vertices that correspond to the symbols in each of the columns of A,

and

- n^2 type 3 vertices that correspond to the rows of A.

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 t 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. The case of a Latin square is just $t = 3$.

We use SageMath [107] to determine if two graphs are isomorphic to decide their equivalence and also to compute the autotopy group of the $(t-2)$-MOLS. For a Latin square, we compute a canonical label with the fixed partition for distinct sets of vertices in the construction of the graph and use the resulting certificate as the main class invariant.

5.5.3.4 Implementation and parameters

- *Environment:* SageMath [107] with GAP [56] for group computations; monitor memory via `psutil` and `GASMAN('collect')`.

- *Parameters:* n, k. For each group G of order kn^2, consider all subgroups H_i of orders kn and K of order k.

- *Enumeration:* Iterate groups by increasing `SmallGroup` index; for each K, then search 3-cliques in Γ_K to fetch all triples (H_1, H_2, H_3).

- *Coset caching:* For each G/K, cache all $(g, H) \mapsto gH$ once and reuse across triples via canonical subgroup identifiers.

- *Outputs:* For each triple, we print G (structure description), the three subgroups, the reduced Latin square, and an associativity verdict. A classification of all non-associative reduced Latin squares into main classes is given. The program generates all associative squares by design so there is no need to save those.

Our scripts are freely available at

`https://github.com/Mansaring/LatinSquareGroup.git`

and we encourage the reader to replicate the results.

5.5.4 Autotopy group data and summary tables

Here we gather the two summary tables cited earlier, namely Tables 5.5.1 and 5.5.2. Table 5.5.1 lists Latin squares of order at most 11 classified by the action of their autotopy groups: for each order we record the number of main classes [63, 85], and then partition these into classes that are isotopic to a Cayley table of a group, classes that are simply transitive but not isotopic to a Cayley table, classes that are transitive but not simply transitive, and classes that are non-transitive. It is known that for any prime p, there are no main classes of Latin squares of order p that are transitive other than the main class containing the cyclic group [36]. This fact simplifies the computations for Latin squares of orders 2, 3, 5, 7, and 11 in Table 5.5.1. For Latin squares of orders 6, 8, 9, and 10, the representatives of main classes, most of which are on the website of Wanless [115], helped in completing Table 5.5.1.

Table 5.5.2 presents representative examples of relatively large sets of MOLS of non-prime power order together with the order of their autotopy groups. For each order we list the size of the MOLS set, the corresponding autotopy group order, and the source of the construction. One noteworthy feature is that the only example in the table in which a proper subset is transitive occurs for the seven MOLS of order 56 [1, 87]. In our computations, we found that a transitive subcollection of six MOLS exists within this family.

Order	Main class	Group	Simply transitive but not group	Transitive but not simply	Non-transitive
2	1	1	0	0	0
3	1	1	0	0	0
4	2	2	0	0	0
5	2	1	0	0	1
6	12	2	1	0	9
7	147	1	0	0	146
8	283,657	5	6	0	283646
9	19,270,853,541	2	0	1	19,270,853,538
10	34,817,397,894,749,939	2	1	8	34,817,397,894,749,928
11	203,602,955,258,288,313,419,609,9	1	0	0	203,602,955,258,288,313,419,609,8

Table 5.5.1: Classification of Latin squares by the action of the autotopy group.

Order n	Size of MOLS set	Autotopy group order	Source
10	2	1	[15]
12	5	12	[23, 65]
14	3	13	[108]
14	4	7	[110]
15	4	14	[2, 102]
15	4	30	[107]
18	3	9	[113]
20	4	19	[9, 109]
21	4	21	[94]
21	5	21	[90]
22	2	21	[25]
22	3	42	[1, 10]
24	7	48	[7]
26	4	21	[44]
28	5	28	[2]
30	4	25	[9]
33	5	33	[8]
34	4	33	[2]
35	5	70	[119]
36	8	324	[7]
38	4	37	[9]
39	5	78	[8]
40	7	160	[5]
42	5	35	[2]
44	5	44	[2]
45	6	135	[3]
46	4	37	[43]
48	8	192	[4]
50	6	43	[43]
51	5	102	[8]
52	5	52	[2]
54	5	45	[2]
55	6	110	[120]
56	7	448	[1, 87]
62	5	54	[2]
75	7	375	[7]
80	9	640	[1, 5]

Table 5.5.2: Autotopy group orders of known MOLS of non-prime power orders.

CHAPTER 6

Conclusion

This dissertation studies mutually orthogonal Latin squares (MOLS) through a blend of computational search, proof certification, and structural group theoretic analysis. A unifying theme across the dissertation is that structure—in the form of transversals, relations in nets, or transitivity under autotopies—can be used both to reduce search spaces and to expose genuine obstructions that are not visible to weaker (pairwise or local) constraints.

At the center of this work is the long-standing problem of determining whether there exist a set of 3 MOLS of order 10. While this dissertation does not resolve the existence of a set of 3 MOLS of order 10, it advances the computational and theoretical toolkit for attacking the problem and related questions, and it clarifies the limitations of prominent structured approaches.

6.1 Summary of results

We give a summary of each of the sets of research results.

6.1.1 A 4×4 subsquare in a set of 3 MOLS(10)

Chapter 3 revisits a structured framework introduced by Myrvold [89] for a hypothetical set of 3 MOLS of order 10 in which one Latin square contains a 4×4 Latin subsquare. Myrvold reduced the search from 28 to 8 pair types by eliminating most of them by arguments involving only two squares at a time. Our central finding is that the remaining cases are not artifacts of an incomplete pairwise analysis: for each of

the surviving pair types, we explicitly construct a transversal representation pair (TRP), and hence an orthogonal pair, satisfying all constraints that arise in the two-square portion of the framework. This provides a concrete explanation for why the Myrvold approach must stop: the obstruction is inherently three-square in nature, so no refinement that only produces reasons for the existence or nonexistence of an orthogonal pair can close the problem within that framework.

Methodologically, our work introduces a practical SAT encoding for TRPs motivated by a concise duality between orthogonality and transversal representations via column-wise composition, yielding an encoding that performs well in practice and is flexible enough to produce explicit witnesses.

6.1.2 Enumeration of 4-nets of order 10 with two relations

Chapter 4 demonstrates how modern SAT solvers can be used not only to search for combinatorial designs, but also to produce trustworthy certificates of nonexistence and exhaustive enumeration. In particular, the chapter recreates and accelerates earlier enumerations of 4-nets(10) with two relations, and complements the computational results with independent verification using proof certificates [46, 58]. A key takeaway is that SAT-based workflows can separate two sources of difficulty: encoding the mathematics correctly, and executing the search. Once the reduction is correct, proofs can be checked independently, making the final results substantially more reliable than custom-written codes for backtracking searches.

Beyond reproducing and validating known results, this chapter illustrates a general template for future investigations in design theory: impose structured constraints that encode mathematically meaningful features like relations, run a solver to either enumerate or refute, and, when possible, emit checkable proofs that turn a large computation into a durable result.

6.1.3 Transitivity and MacNeish's conjecture

Chapter 5 studies sets of MOLS through the lens of transitivity under autotopies. Transitivity is a strong symmetry condition, and it imposes a rich algebraic structure on the associated orthogonal arrays. We develop a framework in which transitive sets of MOLS correspond to group packets, and use this correspondence to derive restrictions on the existence of transitive sets of q MOLS of order n in terms of the prime-power structure of n. In particular, Sylow-theoretic reductions show that, under natural hypotheses on the intersection subgroup K, the size q of a transitive family is bounded by the smallest prime-power divisor of n (in the spirit of the MacNeish-Mann lower bound). This connects transitivity constraints directly to number-theoretic features of the order and supplies obstructions that are invisible without symmetry assumptions.

On the computational side, we develop and apply a construction and classification pipeline that gathers known examples and organizes them by their autotopy group behaviour, producing summary tables that highlight where transitivity has been achieved for non-prime power orders and where gaps remain. We provide examples of Latin squares of the smallest order that are simply transitive but not isotopic to a group, as well as transitive but not simply transitive. We also provide the smallest order n for which there exists a Latin square of order n that is transitive but not simply transitive.

6.2 Future directions

The results in this dissertation suggest several concrete directions for further work.

1. **Three-square encodings in the Myrvold framework.** Since the remaining Myrvold cases admit explicit orthogonal pairs, the next step is to encode (and exploit) constraints that capture triple interactions in a way that still permits effective solving and meaningful symmetry breaking. This includes exploring stronger consistency notions for triples, new invariants of partial orthogonal

arrays, and SAT/CP hybrid approaches that propagate triple constraints more aggressively.

2. **Systematic exploration of structured subproblems for a set of 3 MOLS of order 10.** The 4×4 subsquare setting is only one structured regime. Other regimes—for example, prescribed transversal behaviour, specified subsquare types, or constraints derived from paratopism classes—may yield smaller families of cases that are still representative. A promising goal is to identify a catalogue of structured families that are small enough for exhaustive search with proofs and broad enough that ruling them out would meaningfully constrain the global problem.

3. **Expanding certified enumeration in net and relation problems.** The workflow of Chapter 4 can be extended to larger nets, additional relation types, where classical arguments become unwieldy. Even partial classifications—with proof certificates for excluded cases—could supply new obstructions to the existence of larger sets of MOLS of order 10 via known implications between nets and MOLS.

4. **Transitivity beyond existence: classification and invariants.** For transitive sets of MOLS, it would be valuable to push from existence results to sharper classification theorems: determine which group packets occur for given (n, q), understand the extent to which K and the conjugacy patterns of the H_i determine the main class, and refine the computational pipeline to produce canonical representatives together with group theoretic certificates.

5. **Bridging symmetry and computation in the low-symmetry regime.** One persistent difficulty in a set of 3 MOLS of order 10 is that the objects of interest are known to have little symmetry. Developing a new method that will not exclude solutions and is effective remains crucial. With the little symmetry in the objects of interest, relying on symmetry breaking alone will not solve the problem.

The theory and computations in this dissertation reinforce a central message: progress on difficult existence questions in design theory increasingly comes from tight integration of structural mathematics that isolates the right subproblems and provides genuine obstructions and computational methods that are fast, reproducible, and, when possible, certified by independently checkable proofs. In the case of a set of 3 MOLS of order 10, the results here clarify the limits of a major structured approach, provide new constructive witnesses explaining those limits, and develop tools—both SAT-based and group-theoretic—that can be reused in the next round of attacks on the problem.

REFERENCES

[1] Richard Julian Robert Abel. *On the existence of balanced incomplete block designs and transversal designs*. ProQuest LLC, Ann Arbor, MI, 1995. University of New South Wales (Australia).

[2] Richard Julian Robert Abel. Some $V(12, t)$ vectors and designs from difference and quasi-difference matrices. *The Australasian Journal of Combinatorics*, 40:69–85, 2008.

[3] Richard Julian Robert Abel and Frank E. Bennett. The existence of 2-SOLSSOMs. In *Designs, 2002*, volume 563 of *Math. Appl.*, pages 1–21. Kluwer Acad. Publ., Boston, MA, 2003.

[4] Richard Julian Robert Abel and Nicholas Cavenagh. Concerning eight mutually orthogonal Latin squares. *Journal of Combinatorial Designs*, 15(3):255–261, 2007. doi:10.1002/jcd.20121.

[5] Richard Julian Robert Abel and Yuk W. Cheng. Some new MOLS of order $2^n p$ for p a prime number. *The Australasian Journal of Combinatorics*, 10:175–186, 1994.

[6] Richard Julian Robert Abel, Charles J. Colbourn, and Jeffrey H. Dinitz. Mutually orthogonal Latin squares (MOLS). In *Handbook of Combinatorial Designs*, pages 186–218. Chapman and Hall/CRC, 2006. doi:10.1201/9781420010541.

[7] Richard Julian Robert Abel, Charles J. Colbourn, and Mieczyslaw Wojtas.

Concerning seven and eight mutually orthogonal Latin squares. *Journal of Combinatorial Designs*, 12(2):123–131, 2004. doi:10.1002/jcd.10070.

[8] Richard Julian Robert Abel and Gennian Ge. Some difference matrix constructions and an almost completion for the existence of triplewhist tournaments TWh(v). *European Journal of Combinatorics*, 26(7):1094–1104, 2005. doi:10.1016/j.ejc.2004.04.013.

[9] Richard Julian Robert Abel and Dobromir T. Todorov. Four MOLS of orders 20, 30, 38, and 44. *Journal of Combinatorial Theory. Series A*, 64(1):144–148, 1993. doi:10.1016/0097-3165(93)90093-N.

[10] Richard Julian Robert Abel, Xiafu Zhang, and Hangfu Zhang. Three mutually orthogonal idempotent Latin squares of orders 22 and 26. *Journal of Statistical Planning and Inference*, 51:101–106, 1996. doi:10.1016/0378-3758(95)00073-9.

[11] Wilhelm Ahrens. *Mathematische unterhaltungen und spiele*. BG Teubner, 1918.

[12] Yameen Ajani and Curtis Bright. SAT and lattice reduction for integer factorization. In *Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation*, ISSAC '24, pages 391–399. ACM, July 2024. doi:10.1145/3666000.3669712.

[13] A. A. Albert. Quasigroups. I. *Transactions of the American Mathematical Society*, 54(3):507–519, 1943. doi:10.1090/S0002-9947-1943-0009962-7.

[14] A. A. Albert. Quasigroups. II. *Transactions of the American Mathematical Society*, 55(3):401–419, 1944. doi:10.1090/S0002-9947-1944-0010597-1.

[15] Ian Anderson. *Combinatorial Designs: Construction Methods*. Ellis Horwood, England, 1990.

[16] Gautam Appa, Dimitris Magos, and Ioannis Mourtos. Searching for mutually orthogonal Latin squares via integer and constraint programming. *European*

Journal of Operational Research, 173(2):519–530, September 2006. doi:10.1016/j.ejor.2005.01.048.

[17] Gautam Appa, Ioannis Mourtos, and Dimitris Magos. Integrating constraint and integer programming for the orthogonal Latin squares problem. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, volume 2470 of *Lecture Notes in Computer Science*, pages 17–32. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-46135-3_2.

[18] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In F. Rossi, editor, *Principles and Practice of Constraint Programming – CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-540-45193-8_8.

[19] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. *CaDiCaL 2.0*, page 133–152. Springer Nature Switzerland, 2024. doi:10.1007/978-3-031-65627-9_7.

[20] Armin Biere and Mathias Fleury. Gimsatul, IsaSAT and Kissat entering the SAT competition 2022. *Proc. of SAT Competition: Solver and Benchmark Descriptions, 2022*, pages 10–11, 2022.

[21] Raj Chandra Bose. 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.

[22] Raj Chandra Bose. A note on orthogonal arrays. *Biometrics*, 6(2):179–180, 1950.

[23] Raj Chandra Bose, Indra Mohan Chakravarti, and Donald Ervin Knuth. On methods of constructing sets of mutually orthogonal Latin squares using a computer. I. *Technometrics*, 2(4):507–516, 1960. doi:10.1080/00401706.1960.10489916.

[24] Raj Chandra Bose and Sharadchandra Shankar Shrikhande. 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.

[25] Raj Chandra Bose and Sharadchandra Shankar Shrikhande. On the construction of sets of mutually orthogonal Latin squares and the falsity of a conjecture of Euler. *Transactions of the American Mathematical Society*, 95:191–209, 1960. doi:10.2307/1993286.

[26] Raj Chandra Bose, Sharadchandra Shankar Shrikhande, and Ernest Tilden Parker. 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.

[27] Curtis Bright. The DRAT-trim-t extension of the DRAT-trim checker. https://github.com/curtisbright/drat-trim-t, 2025.

[28] Curtis Bright, Kevin Cheung, Brett Stevens, Dominique Roy, Ilias Kotsireas, and Vijay Ganesh. A nonexistence certificate for projective planes of order ten with weight 15 codewords. *Applicable Algebra in Engineering, Communication and Computing*, 31(3–4):195–213, April 2020. doi:10.1007/s00200-020-00426-y.

[29] Curtis Bright, Kevin K. H. Cheung, Brett Stevens, Ilias Kotsireas, and Vijay Ganesh. Nonexistence certificates for ovals in a projective plane of order ten. In *Combinatorial Algorithms*, pages 97–111. Springer International Publishing, 2020. doi:10.1007/978-3-030-48966-3_8.

[30] Curtis Bright, Kevin K. H. Cheung, Brett Stevens, Ilias Kotsireas, and Vijay Ganesh. A SAT-based resolution of Lam's problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):3669–3676, May 2021. doi:10.1609/aaai.v35i5.16483.

[31] Curtis Bright, Dragomir Ž. Đoković, Ilias Kotsireas, and Vijay Ganesh. A SAT+CAS approach to finding good matrices: New examples and counterexamples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1435–1442, July 2019. doi:10.1609/aaai.v33i01.33011435.

[32] Curtis Bright, Jürgen Gerhard, Ilias Kotsireas, and Vijay Ganesh. Effective problem solving using SAT solvers. In J. Gerhard and I. Kotsireas, editors, *Maple in Mathematics Education and Research*, volume 1125 of *Communications in Computer and Information Science*, pages 205–219. Springer International Publishing, 2020. doi:10.1007/978-3-030-41258-6_15.

[33] Curtis Bright, Amadou Keita, and Brett Stevens. Orthogonal Latin squares of order 10 with two relations: A SAT investigation. *Discrete Mathematics, Algorithms and Applications*, November 2025. To appear. doi:10.1142/s1793830925501563.

[34] Curtis Bright, Amadou Keita, and Brett Stevens. Myrvold's results on orthogonal triples of 10×10 latin squares: A SAT investigation. *The Electronic Journal of Combinatorics*, 33(1):P1.30, 2026. doi:10.37236/13960.

[35] Curtis Bright, Ilias Kotsireas, and Vijay Ganesh. When satisfiability solving meets symbolic computation. *Communications of the ACM*, 65(7):64–72, June 2022. doi:10.1145/3500921.

[36] Joshua M. Browning, Douglas S. Stones, and Ian M. Wanless. Bounds on the number of autotopisms and subsquares of a Latin square. *Combinatorica*, 33(1):11–22, 2013. doi:10.1007/s00493-013-2809-1.

[37] R. H. Bruck. *A Survey of Binary Systems*. Springer, 1958. Reprinted by Dover Publications, 1971. doi:10.2307/3611009.

[38] Richard Hubert Bruck. Finite nets. II. Uniqueness and imbedding. *Pacific Journal of Mathematics*, 13(2):421–457, June 1963. doi:10.2140/pjm.1963.13.421.

[39] Kenneth A. Bush. A generalization of a theorem due to MacNeish. *Annals of Mathematical Statistics*, 23:293–295, 1952. doi:10.1214/aoms/1177729449.

[40] Kenneth A. Bush. Orthogonal arrays of index unity. *The Annals of Mathematical Statistics*, pages 426–434, 1952. doi:10.1214/aoms/1177729387.

[41] Arthur Cayley. On the theory of groups as depending on the symbolic equation $\theta^n = 1$. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 7(42):40–47, 1854. doi:10.1080/14786445408647421.

[42] Arthur Cayley. On the theory of groups. *Proceedings of the London Mathematical Society*, 7(42):126–133, 1877/78. doi:10.1112/PLMS/S1-9.1.126.

[43] K. Chen and L. Zhu. Existence of $v(m, t)$ vectors. *Journal of Statistical Planning and Inference*, 106(1-2):461–471, 2002. doi:10.1016/S0378-3758(02)00228-8.

[44] Charles J. Colbourn. Four MOLS of order 26. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 17:147–148, 1995.

[45] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs*. Chapman and Hall/CRC, 2010. doi:10.1201/9781420010541.

[46] Erin Delisle. 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.

[47] Peter Dukes and Leah Howard. Group divisible designs in MOLS of order ten. *Designs, Codes and Cryptography*, 71(2):283–291, July 2012. doi:10.1007/s10623-012-9729-8.

[48] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Wiley, Hoboken, NJ, 3 edition, 2004.

[49] Judith Egan and Ian M. Wanless. Enumeration of MOLS of small order. *Mathematics of Computation*, 85(298):799–824, July 2015. doi:10.1090/mcom/3010.

[50] Leonhard Euler. Recherches sur une nouvelle espèce de quarrés magiques. *Verhandelingen uitgegeven door het zeeuwsch Genootschap der Wetenschappen te Vlissingen*, pages 85–239, 1782.

[51] Anthony B. Evans. Latin squares without orthogonal mates. *Designs, Codes and Cryptography*, 2006. doi:10.1007/s10623-006-8153-3.

[52] Anthony B. Evans. *Orthogonal Latin Squares Based on Groups*. Springer, 2018. doi:10.1007/978-3-319-94430-2.

[53] Katalin Fazekas, Aina Niemetz, Mathias Preiner, Markus Kirchweger, Stefan Szeider, and Armin Biere. Satisfiability modulo user propagators. *Journal of Artificial Intelligence Research*, 81:989–1017, December 2024. doi:10.1613/jair.1.16163.

[54] Ronald Aylmer Fisher and Frank Yates. The 6×6 Latin squares. *Mathematical Proceedings of the Cambridge Philosophical Society*, 30(4):492–507, 1934. doi:10.1017/S0305004100012731.

[55] Edward Fleisher. *Eulerian squares*. Ph.D. thesis, New York University, 1934.

[56] The GAP Group. *GAP – Groups, Algorithms, and Programming*, 2024. Version 4.13.0. URL: https://www.gap-system.org.

[57] Martin Gardner. *The 2nd Scientific American book of mathematical puzzles & diversions*. Simon and Schuster, 1961.

[58] Michael J. Gill and Ian M. Wanless. Pairs of MOLS of order ten satisfying non-trivial relations. *Designs, Codes and Cryptography*, 91(4):1293–1313, April 2023. doi:10.1007/s10623-022-01149-6.

[59] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean Pythagorean triples problem via cube-and-conquer. In N. Creignou and D. Le Berre, editors, *Theory and Applications of Satisfiability Testing –*

SAT 2016, volume 9710 of *Lecture Notes in Computer Science*, pages 228–245. Springer International Publishing, 2016. doi:10.1007/978-3-319-40970-2_15.

[60] Leah Howard. *Nets of Order $4m + 2$: Linear Dependence and Dimensions of Codes*. Ph.D. thesis, University of Victoria, 2009.

[61] Leah Howard and Wendy Myrvold. A counterexample to Moorhouse's conjecture on the rank of nets. *Bulletin of the Institute of Combinatorics and its Applications*, 60:101–105, 2010.

[62] Pei Huang, Minghao Liu, Cunjing Ge, Feifei Ma, and Jian Zhang. Investigating the existence of orthogonal golf designs via satisfiability testing. In *Proceedings of the 2019 International Symposium on Symbolic and Algebraic Computation*, ISSAC '19, pages 203–210. ACM, July 2019. doi:10.1145/3326229.3326232.

[63] Alexander Hulpke, Petteri Kaski, and Patric R. J. Östergård. The number of Latin squares of order 11. *Mathematics of Computation*, 80(274):1197–1219, 2011. doi:10.1090/S0025-5718-2010-02420-2.

[64] Jiwei Jin, Yiqi Lv, Cunjing Ge, Feifei Ma, and Jian Zhang. Investigating the existence of Costas Latin squares via satisfiability testing. In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing – SAT 2021*, volume 12831 of *Lecture Notes in Computer Science*, pages 270–279. Springer International Publishing, 2021. doi:10.1007/978-3-030-80223-3_19.

[65] Diane M. Johnson, Andrew Lloyd Dulmage, and Nathan S. Mendelsohn. Orthomorphisms of groups and orthogonal Latin squares. i. *Canadian Journal of Mathematics*, 13:356–372, 1961. doi:10.4153/CJM-1961-031-7.

[66] Dieter Jungnickel. On difference matrices and regular Latin squares. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 50:219–231, 1980. doi:10.1007/BF02941430.

[67] Petteri Kaski and Patric R.J. Östergård. *Classification Algorithms for Codes*

and Designs. Algorithms and Computation in Mathematics. Springer-Verlag, 2006. doi:10.1007/3-540-28991-7.

[68] Donald Keedwell and József Dénes. *Latin Squares and their Applications, Second Edition*. Elsevier, 2015. doi:10.1016/c2014-0-03412-0.

[69] Amadou Keita and Ilya Shapiro. Transitive sets of mutually orthogonal Latin squares. *arXiv preprint 2601.22205*, 2026. In submission. doi:10.48550/arXiv.2601.22205.

[70] Stepan Kochemazov, Oleg Zaikin, and Alexander Semenov. The comparison of different SAT encodings for the problem of search for systems of orthogonal Latin squares. In *International Conference Mathematical and Information Technologies-MIT*, pages 155–165, 2016.

[71] Daniel Kotlar. Computing the autotopy group of a Latin square by cycle structure. *Discrete Mathematics*, 331:74–82, 2014. doi:10.1016/j.disc.2014.05.004.

[72] Maurice Kraitchik. *Mathematical Recreations*. Courier Corporation, 2006.

[73] Clement Wing Hong Lam. Opinion: How reliable is a computer-based proof? *The Mathematical Intelligencer*, 12(1):8–12, December 1990. doi:10.1007/bf03023977.

[74] Clement Wing Hong Lam. The search for a finite projective plane of order 10. *The American Mathematical Monthly*, 98(4):305–318, April 1991. doi:10.1080/00029890.1991.12000759.

[75] Clement Wing Hong Lam, Larry Henry Thiel, and Stanley Swiercz. The non-existence of finite projective planes of order 10. *Canadian Journal of Mathematics*, 41(6):1117–1123, December 1989. doi:10.4153/cjm-1989-049-4.

[76] Charles F. Laywine and Gary L. Mullen. *Discrete Mathematics Using Latin Squares*. John Wiley & Sons, 1998.

[77] Runming Lu, Sheng Liu, and Jian Zhang. Searching for doubly self-orthogonal Latin squares. In J. Lee, editor, *Principles and Practice of Constraint Programming – CP 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 538–545. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-23786-7_41.

[78] FeiFei Ma and Jian Zhang. Finding orthogonal Latin squares using finite model searching tools. *Science China Information Sciences*, 56(3):1–9, September 2011. doi:10.1007/s11432-011-4343-3.

[79] Harris F. MacNeish. Euler squares. *Annals of Mathematics*, 23(3):221–227, 1922. doi:10.2307/1967920.

[80] Henry B. Mann. The construction of orthogonal Latin squares. *The Annals of Mathematical Statistics*, 13(4):418–423, 1942. doi:10.1214/aoms/1177731539.

[81] Henry B. Mann. On the construction of sets of orthogonal Latin squares. *The Annals of Mathematical Statistics*, 14(4):401 – 414, 1943. doi:10.1214/aoms/1177731360.

[82] Henry B. Mann. On orthogonal Latin squares. *Bulletin of the American Mathematical Society*, 50:249–257, 1944. doi:10.1090/S0002-9904-1944-08127-5.

[83] João Marques-Silva and Inês Lynce. Towards robust CNF encodings of cardinality constraints. In C. Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, volume 4741 of *Lecture Notes in Computer Science*, pages 483–497. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-74970-7_35.

[84] João P. Marques-Silva and Karem A. Sakallah. GRASP—A new search algorithm for satisfiability. In *Proceedings of International Conference on Computer Aided Design*, ICCAD-96, pages 220–227. IEEE Comput. Soc. Press, 1996. doi:10.1109/iccad.1996.569607.

[85] Brendan D. McKay, Alison Meynert, and Wendy Myrvold. Small Latin squares, quasigroups, and loops. *Journal of Combinatorial Designs*, 15(2):98–119, 2007. doi:10.1002/jcd.20105.

[86] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, January 2014. doi:10.1016/j.jsc.2013.09.003.

[87] William Harold Mills. Some mutually orthogonal Latin squares. *Congr. Numer.*, 19:473–487, 1977.

[88] Eliakim Hastings Moore. Tactical memoranda I-III. *American Journal of Mathematics*, 18(3):264, July 1896. doi:10.2307/2369797.

[89] Wendy Myrvold. Negative results for orthogonal triples of Latin squares of order 10. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 29:95–106, 1999.

[90] A. V. Nazarok. Five pairwise-orthogonal Latin squares of order 21. *Kievskiĭ Gosudarstvennyĭ Universitet. Issledovanie Operatsiĭ i ASU*, 1(35):57–60, 1991.

[91] Donald Norton. Groups of orthogonal row-Latin squares. *Pacific Journal of Mathematics*, 2(3):335–341, September 1952. doi:10.2140/pjm.1952.2.335.

[92] Louis-René Nougier. Essai sur le peuplement préhistorique de la France. *Population (French edition)*, pages 241–274, 1954.

[93] Jacques Ozanam. *Récréations mathématiques et physiques*, volume 4. Claude Jombert, Paris, 1725.

[94] Ernest Tilden Parker. Construction of some sets of mutually orthogonal Latin squares. *Proceedings of the American Mathematical Society*, 10(6):946–949, 1959. doi:10.1090/S0002-9939-1959-0109789-9.

[95] Ernest Tilden Parker. Orthogonal Latin squares. *Proceedings of the National Academy of Sciences*, 45(6):859–862, 1959. doi:10.1073/pnas.45.6.859.

[96] Ernest Tilden Parker. On orthogonal Latin squares. *1960 Institute on Finite Groups*, 6:43–46, 1962. doi:10.1090/pspum/006/0132704.

[97] Julius Petersen. Les 36 officiers. *Annuaire des Mathématiciens*, pages 413–427, 1901.

[98] Calyampudi Radhakrishna Rao. Factorial experiments derivable from combinatorial arrangements of arrays. *Supplement to the Journal of the Royal Statistical Society*, 9:128–139, 1947.

[99] Joseph J. Rotman. *Advanced Modern Algebra*. American Mathematical Society, 3 edition, 2015.

[100] Dominique J. Roy. Confirmation of the non-existence of a projective plane of order 10. Master's thesis, Carleton University, 2011. doi:10.22215/etd/2011-09202.

[101] Noah Rubin, Curtis Bright, Kevin Cheung, and Brett Stevens. Improving integer and constraint programming for Graeco–Latin squares. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, November 2021. doi:10.1109/ictai52525.2021.00096.

[102] Paul J. Schellenberg, G. H. John Van Rees, and Scott A. Vanstone. Four pairwise orthogonal Latin squares of order 15. *Ars Combinatoria*, 6:141–150, 1978.

[103] Carsten Sinz. Towards an optimal CNF encoding of Boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005*, pages 827–831. Springer Berlin Heidelberg, 2005. doi:10.1007/11564751_73.

[104] Jonathan D.H. Smith. *An Introduction to Quasigroups and Their Representations*. Chapman and Hall/CRC, 2006. doi:10.1201/9781420010633.

[105] Peter Ludvig Meidell Sylow. Théorèmes sur les groupes de substitutions. *Mathematische Annalen*, 5:584–594, 1872. doi:10.1007/BF01442913.

[106] Gaston Tarry. Le problème des 36 officiers. *Association Française pour l'Avancement des Sciences: Compte Rendu de la 29^{me} session, Paris 1900*, 2:170–203, 1901.

[107] The Sage Developers. *SageMath, the Sage Mathematics Software System*, 2025. URL: `https://www.sagemath.org`.

[108] Dobromir T. Todorov. Three mutually orthogonal Latin squares of order 14. *Ars Combinatoria*, 20:45–48, 1985.

[109] Dobromir T. Todorov. Four mutually orthogonal Latin squares of order 20. *Ars Combinatoria. A Canadian Journal of Combinatorics*, 27:63–65, 1989.

[110] Dobromir T. Todorov. Four mutually orthogonal Latin squares of order 14. *Journal of Combinatorial Designs*, 20(8):363–367, 2012. `doi:10.1002/jcd.21298`.

[111] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483. Springer, 1983.

[112] Moshe Y. Vardi. Boolean satisfiability: theory and engineering. *Communications of the ACM*, 57(3):5, March 2014. `doi:10.1145/2578043`.

[113] Shinmin Patrick Wang. *On Self-Orthogonal Latin Squares and Partial Transversals of Latin Squares*. Ph.D. thesis, Ohio State University, 1978.

[114] Ian Wanless. Transversals in Latin squares: A survey. In Robin Chapman, editor, *Surveys in Combinatorics 2011*, pages 403–437. Cambridge University Press, June 2011. `doi:10.1017/cbo9781139004114.010`.

[115] Ian M. Wanless. Personal homepage. `https://users.monash.edu.au/~iwanless/`. Accessed: 2026-04-01.

[116] Ian M. Wanless and Bridget S. Webb. The existence of Latin squares without orthogonal mates. *Designs, Codes and Cryptography*, 40(1):131–135, 2006. `doi:10.1007/s10623-006-8168-9`.

[117] Paul Wernicke. Das problem der 36 offiziere. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 19:264–267, 1910.

[118] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, pages 422–429. Springer International Publishing, 2014. doi:10.1007/978-3-319-09284-3_31.

[119] Mieczyslaw Wojtas. Five mutually orthogonal Latin squares of order 35. *Journal of Combinatorial Designs*, 4(2):153–154, 1996. doi:10.1002/(SICI)1520-6610(1996)4:2<153::AID-JCD7>3.0.CO;2-E.

[120] Mieczyslaw Wojtas. Three new constructions of mutually orthogonal Latin squares. *Journal of Combinatorial Designs*, 8(3):218–220, 2000. doi:10.1002/(SICI)1520-6610(2000)8:3<218::AID-JCD7>3.0.CO;2-8.

[121] Oleg Zaikin and Stepan Kochemazov. The search for systems of diagonal Latin squares using the SAT@home project. *International Journal of Open Information Technologies*, 3(11):4–9, 2015.

[122] Oleg Zaikin, Eduard Vatutin, and Curtis Bright. Enumerating extended self-orthogonal diagonal Latin squares of order up to 10. *Journal of Integer Sequences*, 28(7), 2025. Article 25.7.4.

[123] Hantao Zhang. Specifying Latin square problems in propositional logic. In Robert Veroff, editor, *Automated Reasoning and Its Applications: Essays in Honor of Larry Wos*, pages 115–146, Cambridge, Massachusetts, 1997. MIT Press. doi:10.5555/271101.271124.

[124] Hantao Zhang. Combinatorial designs by SAT solvers. In *Handbook of Satisfiability*, pages 819–858. IOS Press, February 2021. doi:10.3233/faia201005.

VITA AUCTORIS

NAME:	Amadou Keita
PLACE OF BIRTH:	Ndorna, Upper Fulladou West District, Central River Region, The Gambia
YEAR OF BIRTH:	1990
EDUCATION:	University of The Gambia, B.Sc.(Honors) in Mathematics (with a minor in Physics), Brikama, West Coast Region, The Gambia, 2014
	African Institute for Mathematical Sciences, M.Sc. in Mathematical Sciences, Accra, Ghana, 2016
	Central European University, M.Sc.(Distinction) in Mathematics and its Applications, Budapest, Hungary, 2019
	African Institute for Mathematical Sciences, M.Sc. in African Masters in Machine Intelligence, Mbour, Senegal, 2022
	University of Windsor, Doctor of Philosophy in Mathematics and Statistics - Mathematics, Windsor, Ontario, Canada, 2026