COMPLETION OF PARTIAL LATIN SQUARES

Benjamin Andrew Burton

Honours Thesis

Department of Mathematics
The University of Queensland
Supervisor: Dr Diane Donovan

Submitted in 1996

Abstract

In this thesis, the problem of completing partial latin squares is examined. In particular, the completion problem for three primary classes of partial latin squares is investigated. First, the theorem of Marshall Hall regarding completions of latin rectangles is discussed. Secondly, a proof of Evans' conjecture is presented, which deals with partial latin squares of order n containing at most $n-1$ entries. Finally, we investigate an open problem regarding completions of k-staggers, which are partial latin squares in which each row and column contains exactly k entries, and each value is used exactly k times.

For this final problem, results are presented for a number of special cases. Computational results are obtained, and on the basis of these, conjectures are drawn. Finally, possible methods of proof for these conjectures are discussed.

Acknowledgments

First and foremost, thanks must go to my supervisor, Diane Donovan, for her fruitful advice, for hearing out my ideas, however bizarre, and for always being there.

Secondly, I must thank Craig Eldershaw for his assistance in computer-related matters, and for causing the honours room much entertainment by bringing down the entire maths department network.

Finally, I would like to thank all the honours students, without whom I might have had this finished some time earlier. But it wouldn't have been nearly as much fun.

Contents

Chapter 1

Introduction

This thesis examines the general problem of completing partial latin squares. A latin square of order n is an $n \times n$ array of cells, filled with elements of $\{1, \ldots, n\}$ in such a fashion that no element appears more than once in the same row or column. A partial latin square is also such an array, but may contain empty cells. The question of completion can then be phrased as follows:

> *Given a partial latin square P, is it possible to fill the empty cells of P so that a latin square is obtained?*

This thesis aims to answer this question for particular classes of partial latin squares P.

Chapter 1 presents a series of preliminary definitions and small results. Following these, a theorem of Philip Hall is presented regarding systems of distinct representatives of sets. Hall's Theorem will be utilised many times throughout this thesis.

Chapter 2 then proves our first completion theorem. Originally proven by Marshall Hall, this shows that any partial latin square in which the first r rows are filled and the remaining rows are empty can be completed. A proof using Philip Hall's theorem is presented, followed by my own proof, which contains within it a direct construction for completing such partial latin squares.

Chapter 3 is devoted to proving our second completion theorem, which is Evans' Conjecture. This states that any partial latin square of order n, in which at most $n-1$ cells have been filled, can always be completed. While the proof is based around that of Smetaniuk, as presented in the literature, it has been considerably elaborated. In particular, the proof of correctness for one of its central constructions is entirely my own.

Finally, Chapter 4 examines k-staggers, which are partial latin squares for which each row and column contains precisely k entries, and each element of $\{1, \ldots, n\}$ appears exactly k times. The material presented in this chapter, except for a small section on orthogonal latin squares, is entirely my own work. After proving that a k-stagger of order n exists whenever $k \leq n$, we examine for which values of k and n it is true that all k-staggers of order n have completions.

A complete solution to this problem is produced for the case $k = 1$. Although the case $k = 2$ is not completely solved, a number of computational results are obtained. From these, a series of conjectures is formed. The chapter then finishes with details on how these conjectures might be proven.

1.1 Preliminary Definitions

1.1.1 Partial Latin Squares

We will let the *natural numbers*, denoted by $\mathbb{N}$, be the set

$$\{1, 2, 3, \ldots\}.$$

Our first task is to define a partial latin square (PLS). Intuitively, we desire for a PLS of order n to be an $n \times n$ array, some of whose locations contain values from the set $\{1, \ldots, n\}$. In addition, a PLS cannot contain the same value more than once in any given row or column. We will now present a formal definition, and then examine the relationship between it and our intuitive idea.

Definitions 1.1.1. Let S be a set. Then S^3 is the set of all ordered triples over S, defined by

$$S^3 = S \times S \times S.$$

Similarly, S^2 is the set of all ordered pairs over S, defined by

$$S^2 = S \times S.$$

Definitions 1.1.2. Let $n \in \mathbb{N}$, and let S be a set of size n. Let $P \subseteq S^3$ have the following property:

- For any $i, j \in S$, P contains:
  - at most one triple of the form (i, j, x);
  - at most one triple of the form (j, x, i);
  - at most one triple of the form (x, i, j).

Then we say P is a *partial latin square*, or *PLS*, of *order* n. We call S the *base set* of P. An element of P is called an *entry* in P.

We will always let $S = \{1, \ldots, n\}$, unless otherwise stated. In particular, for our purposes, this means that two PLSs of the same order can be assumed to have the same base set.

Definitions 1.1.3. Let P be a PLS with base set S. Let $e = (r, c, v)$ be an entry in P. Then r, c and v are called the *row*, *column* and *value* of e respectively.

An ordered pair $(r, c) \in S^2$ is called a *location*, or a *cell*. If $l = (r, c)$ is a location and $e = (r, c, v)$, we say l is the location, or cell, of entry e. We also say that value v *occupies*, or *appears in*, cell l of P. If there is no entry of the form (r, c, v) in P, we say cell l is *empty*.

We can now see how our formal definition relates to our intuitive idea of a PLS. The conditions given in Definitions 1.1.2 correspond to the following concepts:

- Each location contains at most one value;
- Each value occurs at most once in any given row (the *row latin* condition);
- Each value occurs at most once in any given column (the *column latin* condition).

Example 1.1.4. A PLS of order 3 is shown below.

$$\begin{array}{|c|c|c|}\hline 1 & & 3 \\ \hline & 1 & \\ \hline & & 2 \\ \hline\end{array}$$

Remark. The attraction of our formal definition (Definitions 1.1.2) is that it is symmetrical about rows, columns and values. This means, for instance, that any theorem regarding the rows of a PLS immediately implies a corresponding theorem regarding the columns and another regarding the values found within a PLS.

This principle will be referred to as the *principle of symmetry.* An example of its use can be found in Section 2.1.

Definition 1.1.5. A *latin square* is a PLS, with base set S, satisfying the following property:

- For any $i, j \in S$, P contains:
  - exactly one triple of the form (i, j, x);
  - exactly one triple of the form (j, x, i);
  - exactly one triple of the form (x, i, j).

We can again relate this definition to a more intuitive concept of a latin square.

Remark. A latin square is a PLS containing an entry in every possible cell.

Notice also that the definition of a latin square is again symmetrical in rows, columns and values, and thus also allows use of the principle of symmetry.

1.1.2 Extensions and Completions

We now examine the concept of a PLS being a "subsquare" of another PLS.

Definition 1.1.6. Let P, Q be PLSs. Then we say Q is an *extension* of P if:

- P and Q have the same order;
- $P \subseteq Q$ (recalling from Definitions 1.1.2 that both P and Q are subsets of S^3, where S is their common base set).

Intuitively, Q is an extension of P if and only if Q contains all the entries of P.

Example 1.1.7. Q is an extension of P in the example below.

$$P = \begin{array}{|c|c|c|}\hline 1 & & 3 \\ \hline & 1 & \\ \hline & & 2 \\ \hline\end{array}\,, \quad Q = \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 3 & 1 & \\ \hline & 3 & 2 \\ \hline\end{array}\,.$$

Definitions 1.1.8. Let P be a PLS. A *completion* of P is an extension of P that is in fact a latin square.

If P has a completion, we say P is *valid.* If P has no completion, we say P is *invalid.*

Example 1.1.9. In the following example, L is a completion of P. Note that this implies that P is valid.

$$P = \begin{array}{|c|c|c|}\hline 1 & & \\ \hline & 2 & \\ \hline & & 3 \\ \hline\end{array}\,, \quad L = \begin{array}{|c|c|c|}\hline 1 & 3 & 2 \\ \hline 3 & 2 & 1 \\ \hline 2 & 1 & 3 \\ \hline\end{array}\,.$$

Example 1.1.10. Through sufficient case analysis, it can be shown that the following PLS Q has no completion. Note that this implies that Q is invalid.

$$Q = \begin{array}{|c|c|c|}\hline 1 & & 3 \\ \hline & 1 & \\ \hline & & 2 \\ \hline \end{array}$$

So it can be seen that not all partial latin squares have a completion. The remainder of this thesis will be devoted to determining exactly *which* PLSs can be completed.

1.2 Hall's Theorem

A result that will be used throughout this thesis is Hall's Theorem, also called the Marriage Theorem, first proven by Philip Hall in 1935. Before we can present it, however, we need to provide a new definition.

Definitions 1.2.1. Let $n \in \mathbb{N}$ and let $S_1, \ldots, S_n$ be finite sets. A *system of distinct representatives*, or *SDR*, for the S_i is a sequence $\langle s_1, \ldots, s_n \rangle$, where:

- $s_i \in S_i$, for each i;
- the s_i are distinct.

We call s_i the *representative* of S_i, for each i.

Example 1.2.2. If $S_1 = \{3\}$, $S_2 = \{1, 3\}$ and $S_3 = \{1, 2\}$, then a SDR for the S_i is $\langle 3, 1, 2 \rangle$.

Example 1.2.3. Let $S_1 = \{1, 3, 4\}, S_2 = \{2\}, S_3 = \{1\}, S_4 = \{1, 2\}$. Then there is no SDR for these sets.

This can be seen as follows. The representative for S_2 must be 2, and the representative for S_3 must be 1. But, since representatives must be distinct, this leave no possible representative for S_4.

Theorem 1.2.4 (Hall's Theorem)**.** *Let $n \in \mathbb{N}$ and let $S_1, \ldots, S_n$ be finite sets. Then a SDR for these sets exists if and only of the following condition is satisfied:*

- *For all $k \in \{0, \ldots, n\}$ and all choices $S_{i_1}, \ldots, S_{i_k}$ of k sets in our collection (where $i_1, \ldots, i_k$ are distinct), $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$.*

Intuitively, this condition states that the union of any k sets in our collection must contain at least k elements, and that this must be true for every choice of k and every subsequent choice of k sets.

Example 1.2.5. Consider Example 1.2.2. We will examine each choice of k in turn.

- If $k = 0$, the union of 0 sets contains at least 0 elements, as required.
- If $k = 1$, the possible unions are $S_1 = \{3\}$, $S_2 = \{1, 3\}$ and $S_3 = \{1, 2\}$. All of these unions contain at least 1 element, as required.
- If $k = 2$, the possible unions are $S_1 \cup S_2 = \{1, 3\}$, $S_1 \cup S_3 = \{1, 2, 3\}$ and $S_2 \cup S_3 = \{1, 2, 3\}$. All of these unions contain at least 2 elements, as required.

- If $k = 3$, the only possible union is $S_1 \cup S_2 \cup S_3 = \{1, 2, 3\}$, which contains at least 3 elements, as required.

Thus, by Hall's Theorem, a SDR for sets S_1, S_2 and S_3 exists (as was displayed in Example 1.2.2).

Example 1.2.6. Consider Example 1.2.3. Choose $k = 3$, and notice that $S_2 \cup S_3 \cup S_4 = \{1, 2\}$. This is a union of 3 sets, but contains only 2 elements. Thus the required condition is not satisfied, and so by Hall's Theorem, no SDR exists for sets S_1, S_2, S_3 and S_4 (as was noted in Example 1.2.3).

A proof of Hall's Theorem will now be presented, which is an extension of the proof given in [6].

Proof. Our proof will proceed via strong induction on n (note that strong induction in general requires no initial case, such as $n = 1$). Let $r \in \mathbb{N}$. Our inductive hypothesis is that Theorem 1.2.4 is true whenever $0 < n < r$. We must then prove it true for $n = r$.

Throughout this proof, the condition presented in Theorem 1.2.4 (i.e. that the union of any k sets must contain at least k elements) will simply be referred to as "the given condition".

So let $n = r$. First, we shall prove that the existence of a SDR implies the satisfaction of the given condition. Say the SDR $\langle s_1, \ldots, s_n \rangle$ exists for the given sets $S_1, \ldots, S_n$. Now choose any $k \in \{0, \ldots, n\}$ and any k sets $S_{i_1}, \ldots, S_{i_k}$ from our collection (where $i_1, \ldots, i_k$ are distinct). Then, since $s_j \in S_j$ for each j, the union $S_{i_1} \cup \ldots \cup S_{i_k}$ contains all of $s_{i_1}, \ldots, s_{i_k}$. Furthermore, these k representatives are distinct (by definition of a SDR). Thus the union of these k sets contains at least k elements.

Conversely, we shall now assume that the given condition holds, and from this prove the existence of a SDR.

If $n = 1$, the result is trivial. Since the single-set union S_1 contains at least 1 element, say s_1, a SDR is then $\langle s_1 \rangle$.

Thus we may assume $n > 1$. We will now split into two cases.

- Say that, for all $k \in \{1, \ldots, n-1\}$ and all choices of k sets in our collection, the union of these k sets contains at least $k + 1$ elements. Then, in particular, S_1 cannot be empty. So choose some representative $s_1 \in S_1$. It follows that, for all k and all choices of k sets $S_{i_1} \backslash \{s_1\}, \ldots, S_{i_k} \backslash \{s_1\}$ from $S_2 \backslash \{s_1\}, \ldots, S_n \backslash \{s_1\}$, the union of these k sets must contain at least k elements (since we already know that $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k+1$, and at most one element, namely s_1, has been removed).

  Thus we may apply our inductive hypothesis to the $n-1$ sets $S_2 \backslash \{s_1\}, \ldots, S_n \backslash \{s_1\}$, producing distinct representatives $s_2, \ldots, s_n$. Notice then that $s_j \in S_j$ for all $2 \in \{2, \ldots, n\}$, and that none of $s_2, \ldots, s_n$ can be equal to s_1. Hence $\langle s_1, s_2, \ldots, s_n \rangle$ is a SDR for the original sets $S_1, \ldots, S_n$.

- Alternatively, say there is some $k \in \{1, \ldots, n - 1\}$ and some k sets $S_{i_1}, \ldots, S_{i_k}$ whose union contains exactly k elements (from the given condition, it cannot contain fewer). Then apply the inductive hypothesis upon sets $S_{i_1}, \ldots, S_{i_k}$ to produce distinct representatives $s_{i_1}, \ldots, s_{i_k}$.

  Now remove any occurrences of $s_{i_1}, \ldots, s_{i_k}$ from the remaining $n - k$ sets, and call the resultant sets *derived sets*. Say we can find some l and some choice of l derived sets whose union contains fewer than l elements. Then let the original sets from which these were derived be $S_{j_1}, \ldots, S_{j_l}$. It follows then that

$$|S_{j_1} \cup \ldots \cup S_{j_l} \cup \{s_{i_1}, \ldots, s_{i_k}\}| < l + k.$$

However, since $|S_{i_1} \cup \ldots \cup S_{i_k}| = k$ and this union contains all of $s_{i_1}, \ldots, s_{i_k}$, it follows that the union is exactly $\{s_{i_1}, \ldots, s_{i_k}\}$. Thus

$$|S_{j_1} \cup \ldots \cup S_{j_l} \cup S_{i_1} \cup \ldots \cup S_{i_k}| < l + k,$$

contradicting the given condition.

So, for all l and all choices of l derived sets, the union of these l sets contains at least l elements. Thus we can apply the inductive hypothesis to the $n - k$ derived sets, producing a SDR. Let s_j be the representative of the set derived from S_j, for each j (where $j \neq i_m$, for $1 \leq m \leq k$). Notice that there are exactly $n - k$ such values for j.

Notice also, from the definition of the derived sets, that $s_j \in S_j$ for each j (again where $j \neq i_m$, for $1 \leq m \leq k$), and that none of these s_j can be equal to any of the s_{i_m}, for $1 \leq m \leq k$. Hence, combining the representatives s_j with $s_{i_1}, \ldots, s_{i_k}$, we obtain a SDR for the entire collection $S_1, \ldots, S_n$.

So, in all cases, a SDR exists. $\square$

Chapter 2

Completing a Latin Rectangle

This chapter introduces our first completion theorem, first proven by Marshall Hall in 1945, which states that a PLS of order n, with r rows filled and the remaining $n-r$ rows empty, can always be completed.

Two proofs of this theorem will be presented. One of these will be an application of Philip Hall's Theorem, and the other is my own "bare hands" constructive proof.

Note that, throughout this thesis, "Hall's Theorem" will continue to refer to the theorem of Philip Hall (Theorem 1.2.4).

2.1 Theorem Statement

Before the theorem is presented, we need to provide a necessary definition.

Definition 2.1.1. Let $r, n \in \mathbb{N}$, with $r \leq n$. Then an $r \times n$ *latin rectangle* is defined to be a PLS of order n, in which the first r rows are completely filled and the remaining $n-r$ rows are completely empty.

Example 2.1.2. In the following example, P is a 3×4 latin rectangle.

$$P = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 2 & 3 & 4 & 1 \\ \hline 3 & 4 & 1 & 2 \\ \hline & & & \\ \hline \end{array}\,.$$

The completion theorem of Marshall Hall can then be phrased as follows:

Theorem 2.1.3. *Every latin rectangle is valid (i.e. has a completion).*

Example 2.1.4. For instance, the latin square L (shown below) is a completion of P in Example 2.1.2.

$$L = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 2 & 3 & 4 & 1 \\ \hline 3 & 4 & 1 & 2 \\ \hline 4 & 1 & 2 & 3 \\ \hline \end{array}\,.$$

We can use the principle of symmetry (described in Section 1.1.1) to deduce the following immediate corollaries of Theorem 2.1.3:

Corollary 2.1.5. *Let $r, n \in \mathbb{N}$, with $r \leq n$. Let P be a PLS of order n, in which the first r columns are completely filled (i.e. are used in n entries) and the remaining $n - r$ columns are completely empty (i.e. are not used at all).*

Then P is valid.

Example 2.1.6. In the following example, P is a PLS satisfying the given conditions, and L is a completion of P.

$$P = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 5 & & \\ \hline 2 & 3 & 1 & & \\ \hline 3 & 1 & 4 & & \\ \hline 4 & 5 & 3 & & \\ \hline 5 & 4 & 2 & & \\ \hline \end{array}\,, \quad L = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 5 & 4 & 3 \\ \hline 2 & 3 & 1 & 5 & 4 \\ \hline 3 & 1 & 4 & 2 & 5 \\ \hline 4 & 5 & 3 & 1 & 2 \\ \hline 5 & 4 & 2 & 3 & 1 \\ \hline \end{array}\,.$$

Corollary 2.1.7. *Let $r, n \in \mathbb{N}$, with $r \leq n$. Let P be a PLS of order n, in which the first r values are used in n entries and the remaining $n - r$ values are not used at all.*

Then P is valid.

Example 2.1.8. In the following example, P is a PLS satisfying the given conditions, and L is a completion of P.

$$P = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & & 3 & \\ \hline 3 & & 1 & 2 & \\ \hline 2 & & & 1 & 3 \\ \hline & 1 & 3 & & 2 \\ \hline & 3 & 2 & & 1 \\ \hline \end{array}\,, \quad L = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 4 & 3 & 5 \\ \hline 3 & 5 & 1 & 2 & 4 \\ \hline 2 & 4 & 5 & 1 & 3 \\ \hline 4 & 1 & 3 & 5 & 2 \\ \hline 5 & 3 & 2 & 4 & 1 \\ \hline \end{array}\,.$$

We can also derive the following corollary of Theorem 2.1.3, simply by reordering rows within the PLS:

Corollary 2.1.9. *Let $r, n \in \mathbb{N}$, with $r \leq n$. Let P be a PLS of order n, in which some r rows are completely filled and the remaining $n - r$ rows are completely empty.*

Then P is valid.

Example 2.1.10. In the following example, P is a PLS satisfying the given conditions, and L is a completion of P.

$$P = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline & & & \\ \hline 2 & 3 & 4 & 1 \\ \hline & & & \\ \hline \end{array}\,, \quad L = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 3 & 4 & 1 & 2 \\ \hline 2 & 3 & 4 & 1 \\ \hline 4 & 1 & 2 & 3 \\ \hline \end{array}\,.$$

Similar corollaries can be drawn from Corollaries 2.1.5 and 2.1.7.

2.2 Proofs

As described at the beginning of this chapter, two proofs of Theorem 2.1.3 will be presented. The first utilises Hall's Theorem (Theorem 1.2.4), and the second is my own direct constructive proof.

2.2.1 Proof using Hall's Theorem

This proof is an extension of the proof given in [2].

Proof. Let P be an $r \times n$ latin rectangle, where $r \leq n$. We will proceed by induction on t to prove that P can be extended to form a $t \times n$ latin rectangle P_t, where $t = r, r+1, \ldots, n-1, n$.

The case $t = r$ is trivial. The PLS $P_r = P$ is an $r \times n$ latin rectangle that is an extension of P.

Now we proceed to the inductive step. Let $r < t \leq n$, and assume that P can be extended to form a $(t-1) \times n$ latin rectangle P_{t-1}. We must prove that P can also be extended to form a $t \times n$ latin rectangle P_t.

Rows $1, \ldots, (t-1)$ of P_{t-1} are filled, and rows $t, \ldots, n$ are empty. So, if we can show that the tth row of P_{t-1} can be filled with the the integers $1, \ldots, n$ without breaking either the row latin or column latin conditions, then the PLS produced (call it P_t) will be a $t \times n$ rectangle that is an extension of P_{t-1}, and hence of P.

So let U_i be the set of values appearing in column i of P_{t-1}, for $i = 1, \ldots, n$. Then U_i is the set of values that cannot be placed in location (t, i) without breaking the column latin condition (recall that location (t, i) represents the ith cell in row t). So let $S_i = S \backslash U_i$ for all i, where $S = \{1, \ldots, n\}$. Then S_i is the set of values that can be placed in location (t, i) without breaking the column latin condition. For instance, using the following PLS:

$$P_2 = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 3 & 1 & 4 & 2 \\ \hline & & & \\ \hline & & & \\ \hline \end{array},$$

the corresponding S_i are:

$$\begin{aligned} S_1 &= \{2, 4\}, \\ S_2 &= \{3, 4\}, \\ S_3 &= \{1, 2\}, \\ S_4 &= \{1, 3\}. \end{aligned}$$

Now say we can find a SDR $\langle s_1, \ldots, s_n \rangle$ for $S_1, \ldots, S_n$. It follows that this SDR must consist of the integers $1, \ldots, n$ in some order. Then, for each i, we may insert value s_i into location (t, i), thus producing a PLS for which the column latin condition holds, since $s_i \in S_i$ for all i. Furthermore, since all of $1, \ldots, n$ have been inserted into row t, the row latin condition is still true. Thus we have filled in row t of P_{t-1} in the required fashion.

So all that remains is to show that a SDR exists for $S_1, \ldots, S_n$. To the contrary, say there is no SDR. Then there is some k and some choice of sets $S_{i_1}, \ldots, S_{i_k}$ whose union contains less than k elements.

Now, since exactly $t-1$ rows of P_{t-1} are filled, each column of P_{t-1} contains exactly $t-1$ entries. Thus, for each j, we have

$$|S_{i_j}| = |S \backslash U_{i_j}| = n - (t-1) = n - t + 1.$$

So, if we were to write out all the elements of S_{i_1}, followed by the elements of S_{i_2}, and so on up to S_{i_k}, we would have written a total of $k \cdot (n - t + 1)$ elements.

However, since the union $S_{i_1} \cup \ldots \cup S_{i_k}$ contains less than k elements, there must be less than k different values appearing in this written list. Hence, by the pigeonhole principle, at least one value must appear more than $(n - t + 1)$ times, i.e. at least one value must belong to at least $(n - t + 1)$ of the S_i.

So, since $S_i = S \backslash U_i$ for each i, we see that at least one value belongs to less than $t-1$ of the S_i, i.e. it appears in less than $t-1$ columns. However, since exactly $t-1$ rows of the PLS are filled, each with the integers $1, \ldots, n$ exactly once each, every value must appear exactly $t-1$ times in P_{t-1}. Furthermore, since no value can appear twice in the same column, it follows that every value appears in exactly $t-1$ columns. Thus a contradiction has been reached.

So the required SDR exists, the corresponding P_t can be produced, and the induction can be followed through to show that there exists a $n \times n$ latin rectangle P_n which is an extension of P. But an $n \times n$ latin rectangle is simply a latin square, and so the required theorem has been proved. □

Example 2.2.1. We will continue the example presented in the proof. A SDR for the sets $S_1, \ldots, S_4$ is $\langle 4, 3, 2, 1 \rangle$. Thus the first empty row can be filled with these values, giving the latin rectangle

$$P_3 = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 3 & 1 & 4 & 2 \\ \hline 4 & 3 & 2 & 1 \\ \hline & & & \\ \hline \end{array} .$$

A similar procedure is used to fill in the last row, thus producing an entire latin square.

2.2.2 Constructive Proof

The proof presented here is my own constructive proof. As in the previous proof, it involves filling the empty rows, one at a time. However, rather than relying upon existence theorems, a direct construction is provided for filling in these rows. While the proof is therefore longer, the construction presented may be used in situations where an explicit algorithm for completing a latin rectangle is required (for instance, in computational combinatorics).

Construction 2.2.2. Let P be a $(t-1) \times n$ latin rectangle, where $1 \leq t \leq n$. What follows is a construction for filling in row t of P in such a manner that the row latin and column latin conditions are preserved, thus producing a $t \times n$ latin rectangle.

In order to fill in row t, we must place values into cells $(t, 1)$, $(t, 2)$, …, (t, n). We will assume that the first k of these cells (i.e. $(t, 1)$, …, (t, k)) have been filled without violating either the row latin or column latin conditions, and we shall now describe how to fill in cell $(t, k+1)$. By repeatedly following this procedure, the entire row t can be filled in.

Defining value sets:

So, for each i, let U_i be the set of values currently occurring in column i, excluding any value that may have been placed in cell (t, i). Let $S = \{1, \ldots, n\}$, and let $S_i = S \backslash U_i$. Thus S_i is the set of values that may be placed in cell (t, i) without violating the column latin condition.

Consider the following example, which will be referred to throughout our construction.

Our inital latin rectangle is P, as shown below.

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

For the purposes of illustration, we will assume that we have already begun to fill the empty cells of P. Say our current state is as shown below.

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

Here we are in the process of filling row $t = 6$. The first $k = 6$ cells contain values, and our current task is to place a value in cell $(6, 7)$. In this case, the corresponding U_i and S_i are:

$$\begin{aligned}
U_1 &= \{1, 4, 5, 7, 8\}, & S_1 &= \{2, 3, 6\}, \\
U_2 &= \{2, 4, 5, 7, 8\}, & S_2 &= \{1, 3, 6\}, \\
U_3 &= \{3, 5, 6, 7, 8\}, & S_3 &= \{1, 2, 4\}, \\
U_4 &= \{1, 3, 4, 6, 8\}, & S_4 &= \{2, 5, 7\}, \\
U_5 &= \{1, 2, 3, 5, 6\}, & S_5 &= \{4, 7, 8\}, \\
U_6 &= \{1, 2, 3, 6, 7\}, & S_6 &= \{4, 5, 8\}, \\
U_7 &= \{2, 4, 5, 7, 8\}, & S_7 &= \{1, 3, 6\}, \\
U_8 &= \{1, 2, 3, 4, 6\}, & S_8 &= \{5, 7, 8\}.
\end{aligned}$$

In general, note that the currently existing PLS satisfies both the row latin and column latin conditions. So, in order to avoid violating either of these conditions whilst filling row t, the following requirements are necessary and sufficient:

- Any value placed in cell (t, i) must be in S_i, for all i;
- Any two values placed in row t must be different.

These requirements will be referred to as the *insertion conditions*.

Initial construction attempt:

Recall that our aim from this point onwards is simply to fill cell $(t, k + 1)$. If there is an element $x \in S_{k+1}$ that has not already been placed in row t, we can thus insert it into cell $(t, k+1)$. The insertion conditions will be satisfied, and hence our task will be complete.

In our example, $S_{k+1} = S_7 = \{1, 3, 6\}$. However, each of 1, 3 and 6 have already been used in row $t = 6$. So we cannot use a simple insertion as described above. Instead, something more sophisticated will be required.

Constructing index sets:

So assume now that each element of S_{k+1} already appears in row t. In particular, say these elements appear in cells $\{(t, i) \mid i \in I_1\}$, where $I_1 \subseteq \{1, 2, \ldots, n\}$ (so I_1 is the set of columns in which these values appear).

In our example, the elements of S_7 appear in cells $(6, 1)$, $(6, 2)$ and $(6, 3)$. Thus $I_1 = \{1, 2, 3\}$.

In general, if $I \subseteq \{1, 2, \ldots, n\}$, we will let $\epsilon(I)$ denote the set $\{\epsilon(t, i) \mid i \in I\}$, where $\epsilon(x, y)$ denotes the value in cell (x, y). So in our example, where $I_1 = \{1, 2, 3\}$, we have $\epsilon(I_1) = \{3, 6, 1\}$. In fact, in general, by definition of I_1, we have $\epsilon(I_1) = S_{k+1}$. Note also that $\epsilon(I \cup J) = \epsilon(I) \cup \epsilon(J)$, for all sets $I, J \subseteq \{1, 2, \ldots, n\}$. This then implies that, for all sets $I, J \subseteq \{1, 2, \ldots, n\}$, if $I \subseteq J$, we have $\epsilon(I) \subseteq \epsilon(J)$.

Furthermore, if $I \subseteq \{1, 2, \ldots, n\}$, we will let S_I denote the set $\bigcup_{i \in I} S_i$. So in our example, where $I_1 = \{1, 2, 3\}$, we have

$$S_{I_1} = S_1 \cup S_2 \cup S_3 = \{2, 3, 6\} \cup \{1, 3, 6\} \cup \{1, 2, 4\} = \{1, 2, 3, 4, 6\}.$$

In general, our insertion requirements imply that, for any $I \subseteq \{1, 2, \ldots, n\}$, we have $\epsilon(I) \subseteq S_I$, since the entry in cell (t, i) must be in S_i, for all $i \in I$. Note also that $S_{I \cup J} = S_I \cup S_J$, for all $I, J \subseteq \{1, 2, \ldots, n\}$.

In the general case, we continue to construct *index sets* $I_2, I_3, \ldots$ as follows. We will later show that this construction must terminate at some point.

1. Say the last set constructed was I_m (so, to begin with, $m = 1$).

2. If S_{I_m} contains an element x that has not yet been placed in row t, we say the sequence $\langle I_1, \ldots, I_m \rangle$ is *ecstatic*, and we stop constructing index sets.

3. Otherwise, consider the set $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$. We will prove shortly that this set cannot be empty. Since we know that every element of S_{I_m} has been placed in row t, we know that every element of $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$ has been placed in row t. In particular, say they appear in locations $\{(t, i) \mid i \in I_{m+1}\}$ (so I_{m+1} is the set of columns in which these values appear). This then defines set I_{m+1}, and we return to step 1.

   Note then that every element of S_{I_m} occurs either in $\epsilon(I_1 \cup \ldots \cup I_m)$ or in $\epsilon(I_{m+1})$. So

   $$S_{I_m} \subseteq \epsilon(I_1 \cup \ldots \cup I_m) \cup \epsilon(I_{m+1}) = \epsilon(I_1 \cup \ldots \cup I_m \cup I_{m+1}).$$

   Note also that I_{m+1} is non-empty, since $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$ is non-empty. Furthermore, say we can find some x satisfying $x \in I_{m+1}$ and $x \in I_j$, for some $j \leq m$. Then, since $x \in I_j$, the value in cell (t, x) belongs to $\epsilon(I_1 \cup \ldots \cup I_m)$. However, since $x \in I_{m+1}$, the value in cell (t, x) belongs to $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$. But no value can belong to both of these sets. Thus I_{m+1} is disjoint from each of $I_1, \ldots, I_m$.

Proof that $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$ is non-empty:

Recall that we promised to show that the set $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$ cannot be empty. We will now prove this claim. Say, on the other hand, that this set is in fact empty. Then $S_{I_m} \subseteq \epsilon(I_1 \cup \ldots \cup I_m)$. However, recall also that, for each $j \leq m-1$, $S_{I_j} \subseteq \epsilon(I_1 \cup \ldots \cup I_{j+1})$. This in turn gives

$$S_{I_j} \subseteq \epsilon(I_1 \cup \ldots \cup I_{j+1}) \subseteq \epsilon(I_1 \cup \ldots \cup I_m).$$

Combining this with the previous note, we see that $S_{I_j} \subseteq \epsilon(I_1 \cup \ldots \cup I_m)$, for all $j \in \{1, \ldots, n\}$. Thus

$$S_{I_1 \cup \ldots \cup I_m} \subseteq \epsilon(I_1 \cup \ldots I_m).$$

So let $I_0 = I_1 \cup \ldots \cup I_m$. Then $S_{I_0} \subseteq \epsilon(I_0)$. However, it was noted earlier that $\epsilon(I) \subseteq S_I$, for any I. Thus $S_{I_0} = \epsilon(I_0)$.

Recall now that $S_{k+1} = \epsilon(I_1)$. So, in fact, we have $S_{I_0} \cup S_{k+1} = \epsilon(I_0) \cup \epsilon(I_1) = \epsilon(I_0)$, since $I_1 \subseteq I_0$. Thus $S_{I_0 \cup \{k+1\}} = \epsilon(I_0)$.

Let us now examine what we have. I_0 is a non-empty set of columns, all of which contain an entry in row t (by definition of the index sets I_j). Let $s = |I_0|$. Furthermore, column $(k+1)$ does not contain an entry in row t. Hence $|I_0 \cup \{k+1\}| = s+1$. Define $I' = I_0 \cup \{k+1\}$.

Note also that, since $\epsilon(I_0)$ is the set of all values appearing in row t and columns in I_0, we have $|\epsilon(I_0)| = |I_0| = s$. Hence $|S_{I'}| = |\epsilon(I_0)| = s$, and $|I'| = s+1$.

So we have found $s+1$ sets from the collection $S_1, \ldots, S_n$ (namely $\{S_i \mid i \in I'\}$), whose union contains only s elements. Let these sets be $S_{i_1}, \ldots, S_{i_{s+1}}$. Since the union contains only s elements, there are $n-s$ values belonging to none of the S_{i_j}. Hence these $n-s$ values occur in each of columns $i_1, \ldots, i_{s+1}$, even when excluding row t, and so account for at least $(n-s)(s+1)$ of the entries in these $s+1$ columns. Any entries in row t are to be excluded until further notice. Thus the remaining s values can account for at most $(t-1)(s+1) - (n-s)(s+1) = (t-n+s-1)(s+1)$ of the entries in these $s+1$ columns, since each column contains a total of $t-1$ entries. Now

$$\begin{aligned}(t-n+s-1)(s+1) &= ts + t + s^2 + s - ns - n - s - 1 \\ &= s(s-n+t) + (t-n-1) \\ &< s(s-n+t),\end{aligned}$$

since $t \leq n$. So these s values account for less than $s(s-n+t)$ entries in the columns contained in I_0, and so at least one of these values occurs less than $(s-n+t)$ times in these columns. Let this value be x.

Furthermore, in the remaining $(n-(s+1)) = (n-s-1)$ columns not contained in I_0, x can appear at most $n-s-1$ times, since x can appear in each column at most once. Thus, in total, x appears in less than $(s-n+t) + (n-s-1) = t-1$ columns.

However, since each of the $(t-1)$ filled rows of P_{t-1} contains each value exactly once, x must appear exactly $(t-1)$ times. So, since it can appear in each column at most once, x must appear in exactly $(t-1)$ columns. We have thus arrived at a contradiction.

Thus $S_{I_m} \backslash \epsilon(I_1 \cup \ldots \cup I_m)$ is non-empty.

Proof that construction terminates:

Since the sets $I_1, I_2, \ldots$ are non-empty and disjoint, and their elements all belong to the finite set $\{1, \ldots, n\}$, it follows that there can only be finitely many such index sets. Thus our index set constructions must terminate due to the existence of an ecstatic sequence $\langle I_1, \ldots, I_m \rangle$.

Illustration of index sets:

The construction of index sets is now illustrated for our example. Recall that $I_1 = \{1, 2, 3\}$ and $S_{I_1} = \{1, 2, 3, 4, 6\}$. All of these values occur in row $t = 6$, so we must continue to construct index sets. Recall also that $\epsilon(I_1) = \{1, 3, 6\}$. Thus $S_{I_1} \backslash \epsilon(I_1) = \{2, 4\}$. These values occur in columns 4 and 5, so $I_2 = \{4, 5\}$.

Now we have

$$S_{I_2} = S_4 \cup S_5 = \{2, 5, 7\} \cup \{4, 7, 8\} = \{2, 4, 5, 7, 8\}.$$

This contains the value 7, which has not yet been placed in row $t = 6$. Thus the sequence $\langle I_1, I_2 \rangle$ is ecstatic, and we stop constructing index sets.

Finding columns:

In general, assume we have an ecstatic sequence $\langle I_1, \ldots, I_m \rangle$. We will now describe how to fill cell $(t, k + 1)$.

Since this sequence is ecstatic, there is some $x_m \in S_{I_m}$ that does not already appear in row t. Then $x_m \in S_{c_m}$, for some $c_m \in I_m$. Let x_{m-1} be the entry in cell (t, c_m).

Now, from the definition of I_m, we have $x_{m-1} \in S_{I_{m-1}}$. So $x_{m-1} \in S_{c_{m-1}}$, for some $c_{m-1} \in I_{m-1}$. Let x_{m-2} be the entry in cell (t, c_{m-1}).

In general, for each $i \in \{m - 1, m - 2, \ldots, 1\}$, we notice that, from the definition of I_{i+1}, we have $x_i \in S_{I_i}$. So $x_i \in S_{c_i}$, for some $c_i \in I_i$. Then let x_{i-1} be the entry in cell (t, c_i). We continue this procedure until c_1 and x_0 have been evaluated. Finally, by definition of I_1, we know $x_0 \in S_{k+1}$.

This procedure will be illustrated using our example. Recall that we found an ecstatic sequence $\langle I_1, I_2 \rangle$, where $I_1 = \{1, 2, 3\}$ and $I_4 = \{4, 5\}$. We also found the value $7 \in S_{I_2}$ that has not yet been placed in row $t = 6$. So let $x_2 = 7$. Since 7 belongs to both S_4 and S_5, we have a choice of columns to use as c_2. We shall choose $c_2 = 5$. Then the entry in cell $(6, 5)$ is 4, so we define $x_1 = 4$.

Now we need to find column $c_1 \in I_1 = \{1, 2, 3\}$ satisfying $x_1 = 4 \in S_{c_1}$. In this case, $4 \in S_3$. So we define $c_1 = 3$. Finally, the entry in cell $(6, 3)$ is 1, so we let $x_0 = 1$.

Filling cells:

At this stage, we are ready to fill in our cells! The procedure is as follows:

- For $i = 1, 2, \ldots, m$, remove the value in cell (t, c_i) and replace it with the value x_i.
- Place value x_0 in cell $(t, k + 1)$.

This then fills cell $(t, k + 1)$ as required, without emptying any of the cells already filled.

Since $x_i \in S_i$ for $i = 1, \ldots, m$ and $x_0 \in S_{k+1}$, the column latin condition is not violated by our construction. Furthermore, the only new value placed in row t is x_m, which (by definition) does not already occur in row t. The remaining values are simply shifted around. In general, for $i = 0, 1, \ldots, (m-1)$, the value x_i is removed from cell (t, c_{i+1}) and placed into cell (t, c_i). Thus any two values appearing in row t are different. Hence the insertion requirements are satisfied, and so the resulting PLS satisfies both the row latin and column latin conditions.

Furthermore, the PLS obtained is still an extension of our original latin rectangle, since the cells that were altered (namely (t, c_1), …, (t, c_m)) were empty in our original latin rectangle P.

Consider our example. Recall that

$$\begin{aligned} x_2 &= 7, & c_2 &= 5, \\ x_1 &= 4, & c_1 &= 3, \\ x_0 &= 1. \end{aligned}$$

So we replace cell $(6, 5)$ with 7 and cell $(6, 3)$ with 4. Finally, the value 1 is inserted into the empty cell $(t, k + 1) = (6, 7)$. The resulting PLS is shown below.

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

This procedure for filling cell $(t, k + 1)$ can be reiterated until the entire row is filled. Then, in turn, each row can be filled until the entire PLS has been filled. This then produces a completion of our original latin rectangle.

In our example, we have now filled cells $(6, 1)$, …, $(6, 7)$, and so we let $k = 7$. Our next task is then to fill cell $(t, k + 1) = (6, 8)$. Thankfully, this is easier than filling the previous cell, since S_8 contains the value 8, which does not yet appear in row $t = 6$. Thus we simply insert value 8 into cell $(6, 8)$. This then completes row 6. So we now let $t = 7$ and $k = 0$, and the procedure is continued. A possible final completion of our original latin rectangle P is shown below.

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

The proof of Theorem 2.1.3 is now as follows:

Proof. Let P be an $r \times n$ latin rectangle. If $r < n$, then Construction 2.2.2 can be used to extend P to an $(r + 1) \times n$ latin rectangle. If $r + 1 < n$, we use Construction 2.2.2 again to produce an $(r + 2)$ latin rectangle. This procedure is continued until an $n \times n$ latin rectangle has been produced. This is then a latin square that is a completion of P, and so P is valid. $\square$

Chapter 3

Evans' Conjecture

This chapter is devoted to a second completion theorem, known as Evans' Conjecture. Originally proposed by Trevor Evans in 1960, this theorem states that any PLS of order n containing at most $n-1$ entries is valid (i.e. can be completed). However, at the time it remained unproven (hence the name "Evans's *Conjecture*"). In the years following, many partial results were produced, one of which will be presented in this chapter (Theorem 3.2.1). However, it was not until 1981 that a complete proof was provided, in this case by Bohdan Smetaniuk [5].

We will begin by formally presenting Evans' Conjecture in the form of a conjecture. Then, after producing some brief illustrations, we will develop a certain amount of necessary background theory. Following this, Smetaniuk's proof of Evans' Conjecture will be discussed.

3.1 Outline

Evans' Conjecture is then as follows.

Conjecture 3.1.1. *Let P be a PLS of order n. If P contains at most $n-1$ entries, then P is valid.*

Example 3.1.2. In the illustration below, P is a PLS of order 5 containing 4 entries, and L is a completion of P.

$$P = \begin{array}{|c|c|c|c|c|}\hline 1 & & & & \\ \hline & & 3 & & \\ \hline & & & & \\ \hline & 3 & 4 & & \\ \hline & & & & \\ \hline \end{array}\,, \quad L = \begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 5 & 3 & 2 \\ \hline 5 & 2 & 3 & 1 & 4 \\ \hline 3 & 1 & 2 & 4 & 5 \\ \hline 2 & 3 & 4 & 5 & 1 \\ \hline 4 & 5 & 1 & 2 & 3 \\ \hline \end{array}\,.$$

Notice also that the upper bound of $n-1$ entries is, in a sense, the best possible, as illustrated by the following result, based upon an example from [5].

Lemma 3.1.3. *Let $n \in \mathbb{N}$, where $n > 1$. Then there exists a PLS of order n, containing exactly n entries, which is invalid (i.e. has no completion).*

Proof. Let P be the PLS constructed as follows:

- The value 1 is placed in cells $(1,1)$, …, $(n-1, n-1)$;
- The value 2 is placed in cell (n, n);

- All other cells remain empty.

The corresponding PLS for $n = 4$ is shown below.

1			
	1		
		1	
			2

We can show that P has no completion as follows. Let L be a completion of P. Then the value 1 must appear once in each row of L, and so in particular it must appear in row n of L. Furthermore, it cannot occur in any of the cells $(n, 1)$, …, $(n, n-1)$, since the value 1 already appears in cell (i, i), for each $i \in \{1, \ldots, n-1\}$. in columns $1, \ldots, n-1$. Thus the value 1 must appear in cell (n, n), and so we have a contradiction (since 2 has already been placed in cell (n, n)).

Hence P is a PLS of order n, containing exactly n entries, and is invalid. □

3.2 A Partial Result

The following result is a special case of Evans' Conjecture, and was proven by Curt Lindner in 1970, well before a proof for Evans' Conjecture was known. This result will come into use later, during the discussion of Smetaniuk's proof.

Theorem 3.2.1. *Let P be a PLS of order n, containing at most $n-1$ entries. If these entries lie in at most $n/2$ of the rows of P, then P is valid.*

The following proof is an extension of that given in [4].

Proof. This proof will use similar techniques to those used in the proof of Theorem 2.1.3 presented in Section 2.2.1.

For the time being, we will assume that P contains exactly $n-1$ entries. The case in which P contains fewer entries will be dealt with at the end of this proof.

Permuting Rows

Let P be a PLS of order n, containing at most $n-1$ entries, and say these entries occur in at most $n/2$ distinct rows of P. In particular, say exactly m of the rows of P contain entries (where $m \leq (n/2)$). Then it is possible to permute the rows of P, using some permutation α, in order to obtain a new PLS Q that satisfies the following properties:

- Rows $1, 2, \ldots, m$ of Q contain entries, and rows $m+1, m+2, \ldots, n$ of Q are empty.
- Let r_i denote the number of entries in row i of Q, where $i \in \{1, \ldots, m\}$. Then $r_1 \geq r_2 \geq \ldots \geq r_m$.

For instance, let P be the following PLS of order $n = 6$:

$P =$

		1	4		
			5		
	3			4	

.

Note that P contains $5 \leq 6-1$ entries, and that these entries lie within $m = 3 \leq (6/2)$ distinct rows of P. So, using the permutation[1] $\alpha = (1\ 4\ 3\ 5\ 2)$, we can rearrange the rows of P to form Q as shown below.

$$Q = \begin{array}{|c|c|c|c|c|c|}\hline & & 1 & 4 & & \\ \hline & 3 & & & 4 & \\ \hline & & & 5 & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}\ .$$

It then follows that the values of r_i for $i \in \{1, \ldots, m = 3\}$ are as follows:

$$\begin{aligned} r_1 &= 2; \\ r_2 &= 2; \\ r_3 &= 1. \end{aligned}$$

In general, we will show that the rows of Q can be filled, one at a time, until a completion L' of Q is produced.

Filling the first row:

First, we will consider row 1 of Q, which contains $n - r_1$ empty cells. Without loss of generality, let these be cells $(1,1)$, $(1,2)$, ..., $(1, n-r_1)$.

Now let R_1 be the set of values already appearing in row 1. Furthermore, for all $i \in \{1, \ldots, n-r_1\}$, let C_i be the set of values already appearing in column i. Then C_i and R_1 together represent the set of values that cannot be placed in cell $(1,i)$ without violating either the row latin or column latin condition. So, for each $i \in \{1, \ldots, n-1\}$, let $S_i = S \backslash (C_i \cup R_1)$, where $S = \{1, \ldots, n\}$. Then S_i represents the set of values that may be placed in cell $(1,i)$ without violating either the row latin or column latin condition.

Consider the example below, obtained from Q above by permuting columns so that cells $(1,1)$, ..., $(1,4) = (1, 6-r_1)$ are empty.

$$\begin{array}{|c|c|c|c|c|c|}\hline & & & & 1 & 4 \\ \hline & 3 & & 4 & & \\ \hline & & & & & 5 \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}$$

In this case, $R_1 = \{1, 4\}$. The corresponding C_i and S_i are then:

$$\begin{aligned} C_1 = \{\}, &\qquad S_1 = \{2,3,5,6\}, \\ C_2 = \{3\}, &\qquad S_2 = \{2,5,6\}, \\ C_3 = \{\}, &\qquad S_3 = \{2,3,5,6\}, \\ C_4 = \{4\}, &\qquad S_4 = \{2,3,5,6\}. \end{aligned}$$

In general, if we can find a SDR $\langle s_1, \ldots, s_{n-r_1} \rangle$ for $S_1, \ldots, S_{n-r_1}$, we can place value s_i in cell $(1,i)$, for each $i \in \{1, \ldots, n-r_1\}$. Let the resulting PLS be Q_1. Because $s_i \in S_i$ for

[1] Throughout this thesis, all permutations will be written using cycle notation.

each i, the column latin condition holds for Q_1. Furthermore, none of $s_1, \ldots, s_{n-r_1}$ will be equal to any of the values previously existing in row 1. Finally, because $s_1, \ldots, s_{n-r_1}$ are distinct, none of these values will be equal to each other. Thus row 1 will contain n distinct values, and the row latin condition for Q_1 will also hold. So Q_1 will be an extension of Q in which the first row has been completely filled (and the remaining cells have been left untouched).

In the above example, a SDR for $S_1, \ldots, S_4$ is $\langle 3, 6, 5, 2 \rangle$. So the first row can then be completed as shown below.

$$Q_1 = \begin{array}{|c|c|c|c|c|c|}\hline 3 & 6 & 5 & 2 & 1 & 4 \\ \hline & 3 & & 4 & & \\ \hline & & & & & 5 \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}\ .$$

We will now show in general that a SDR for $S_1, \ldots, S_{n-r_1}$ must exist. This will be proven using Hall's Theorem.

Choose any $k \in \{0, \ldots, n-r_1\}$ and any k sets from the above collection. Let these be $S_{i_1}, \ldots, S_{i_k}$. If $k = 0$, the union of 0 sets contains at least 0 elements, as required.

So let $1 \leq k \leq n - r_1$. We will consider two cases.

- Say one of the sets $C_{i_1}, \ldots, C_{i_k}$ is empty. Let this set be C_{i_j}. Then $S_{i_j} = S \backslash R_1$, and so $|S_{i_j}| = n - r_1 \geq k$. Hence $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$.

- Otherwise, none of $C_{i_1}, \ldots, C_{i_k}$ are empty. So each of columns $i_2, \ldots, i_k$ contains at least one entry. Furthermore, row 1 contains another r_1 entries (these occur in columns $n - r_1 + 1, \ldots, n$ respectively).

  Hence, since Q contains at most $n-1$ entries, column C_{i_1} can contain at most $(n-1) - (k-1) - r_1 = n - k - r_1$ entries. Since $S_{i_1} = S \backslash (C_{i_1} \cup R_1)$, it follows that

  $$|S_{i_1}| \geq |S| - |C_{i_1}| - |R_1| \geq n - (n - k - r_1) - r_1 = k.$$

  So again we have $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$.

Thus, in either case, $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$. So, by Hall's Theorem, sets $S_1, \ldots, S_{n-r_1}$ have a SDR, as required.

Thus the first row of Q can be filled, as described above, to produce the PLS Q_1.

Filling rows $2, \ldots, m$:

Assume that we have filled rows $1, \ldots, t$ of Q, where $1 \leq t \leq m-1$, to produce the extension Q_t of Q. We will show that row $(t+1)$ can now be filled, producing a new extension Q_{t+1} of Q.

To begin with, we will show that

$$r_1 + r_2 + \ldots + r_t \geq 2t. \tag{i}$$

Otherwise, say this is not the case. Then

$$r_1 + r_2 + \ldots + r_t \leq 2t - 1,$$

and so

$$(n-1) - (r_1 + r_2 + \ldots + r_t) \geq (n-1) - (2t-1) = n - 2t \geq 2m - 2t, \qquad \text{(ii)}$$

since $m \leq (n/2)$. However, the left hand side of this equation represents the number of entries appearing in rows $t+1, t+2, \ldots, m$. There are $(m-t)$ such rows (note that $m - t > 0$, since $t \leq m-1$). Since at least $2(m-t)$ entries appear in these rows, one of these rows must contain at least two entries.

Since $r_1 \geq r_2 \geq \ldots \geq r_m$, we can then deduce that rows $1, 2, \ldots, t$ must each contain at least two entries. Thus $r_1 + r_2 + \ldots + r_t \geq 2t$. But this is (i), which was assumed to be false. Hence a contradiction arises.

So equation (i) is true.

Recall now that we are extending Q_t by filling row $(t+1)$, which already contains r_{t+1} elements. Without loss of generality, let the empty cells of row $(t+1)$ be $(t+1, 1)$, ..., $(t+1, n-r_{t+1})$. Then define the following sets:

- Let R_{t+1} be the set of all values already appearing in row $(t+1)$ of Q_t.
- For $i = 1, \ldots, n - r_{t+1}$, let C_i be the set of all values already appearing in column i of Q_t, excluding those in rows $1, \ldots, t$. Specifically, C_i contains all values appearing in any of the cells $(t+1, i)$, $(t+2, i)$, ..., (m, i).
- For $i = 1, \ldots, n - r_{t+1}$, let T_i be the set of all values already appearing in column i of Q, but including only those in rows $1, \ldots, t$. Specifically, T_i contains all values appearing in cells $(1, i)$, $(2, i)$, ..., (t, i).
- For $i = 1, \ldots, n - r_{t+1}$, let $S_i = S \backslash (C_i \cup T_i \cup R_{t+1})$, where $S = \{1, \ldots, n\}$.

Then, for each i, sets C_i, T_i and R_{t+1} together represent the values that cannot be placed in cell $(t+1, i)$ without violating either the row latin or column latin condition. So S_i is the set of values that can be placed in cell $(t+1, i)$ without violating either of these conditions.

Consider the following example, obtained from Q_1 above by rearranging columns so that the empty cells in row $t+1 = 2$ are $(2, 1)$, ..., $(2, 4) = (2, 6 - r_2)$.

3	1	5	4	6	2
				3	4
			5		

Then we have $R_2 = \{3, 4\}$, and the remaining sets are:

$$\begin{array}{lll} C_1 = \{\}, & T_1 = \{3\}, & S_1 = \{1, 2, 5, 6\}; \\ C_2 = \{\}, & T_2 = \{1\}, & S_2 = \{2, 5, 6\}; \\ C_3 = \{\}, & T_3 = \{5\}, & S_3 = \{1, 2, 6\}; \\ C_4 = \{5\}, & T_4 = \{4\}, & S_4 = \{1, 2, 6\}. \end{array}$$

As before, we will aim to find a SDR $\langle s_1, \ldots, s_{n-r_{t+1}} \rangle$ for sets $S_1, \ldots, S_{n-r_{t+1}}$. If such a SDR can be found, we can place value s_i in cell $(t+1, i)$, for all $i \in \{1, \ldots, n - r_{t+1}\}$. Let the resulting PLS be Q_{t+1}.

Again, because $s_i \in S_i$, the column latin condition of Q_{t+1} is still satisfied. Furthermore, none of $s_1, \ldots, s_{n-r_{t+1}}$ will be equal to any of the values previously existing in row $(t+1)$. Also, since the representatives $s_1, \ldots, s_{n-r_{t+1}}$ are distinct, none of these values will be equal to each other. Thus the row latin condition of Q_{t+1} will also be satisfied. Hence Q_{t+1} will be an extension of Q in which the first $(t+1)$ rows are filled, and the remaining cells have been left untouched.

In the above example, a SDR for sets $S_1, \ldots, S_4$ is $\langle 1, 5, 6, 2 \rangle$. We can thus complete the second row as shown below.

3	1	5	4	6	2
1	5	6	2	3	4
			5		

We will now show that, in general, a SDR must exist for sets $S_1, \ldots, S_{n-r_{t+1}}$. Again, Hall's Theorem will be called upon.

Choose any $k \in \{0, \ldots, n - r_{t+1}\}$ and any k sets from the above collection. Let these be $S_{i_1}, \ldots, S_{i_k}$. If $k = 0$, the union of 0 sets contains at least 0 elements, as required.

Now say $k = 1$ and set S_{i_1} is chosen. If $r_{t+1} \geq (n/2)$, this would imply $r_1 \geq r_{t+1} \geq (n/2)$. Hence our original PLS P would have contained at least $r_1 + r_{t+1} \geq n$ entries, which is a contradiction. So $r_{t+1} < (n/2)$.

Furthermore, the total number of entries already existing in column i can be at most $m - 1$, since cell $(t+1, i)$ is empty. Thus $|C_i \cup T_i| \leq m - 1 \leq (n/2) - 1$. We can then deduce that

$$\begin{aligned}
|S_i| &= |S \backslash (C_i \cup T_i \cup R_{t+1})| \\
&\geq |S| - |C_i \cup T_i| - |R_{t+1}| \\
&= n - |C_i \cup T_i| - r_{t+1} \\
&> n - [(n/2) - 1] - (n/2) \\
&= 1.
\end{aligned}$$

Thus $|S_i| \geq 1 = k$, as required..

So now say $2 \leq k \leq n - r_{t+1}$. Note that $|R_{t+1}| = r_{t+1}$ and that $|T_i| = t$, for all i. We will consider two cases.

1. Say $2 \leq k \leq n - t - r_{t+1}$. Two sub-cases will now be examined.

   (a) Say one of $C_{i_1}, \ldots, C_{i_k}$ is empty. Let this set be C_{i_j}. Then $S_{i_j} = S \backslash (T_{i_j} \cup R_{t+1})$, and so
   $$|S_{i_j}| \geq |S| - |T_{i_j}| - |R_{t+1}| = n - t - r_{t+1} \geq k,$$
   by our case definition. Hence $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$.

   (b) Otherwise, none of $C_{i_1}, \ldots, C_{i_k}$ are empty. Consider the entries in Q_t appearing below row t. There are at least $|C_1| + \ldots + |C_{n-r_{t+1}}| + |R_{t+1}|$ such entries, which can be seen from the definitions of sets C_i and R_{r+1}. However, we also know that there are exactly $r_{t+1} + \ldots + r_m$ such entries. Thus
   $$\begin{aligned}
   |C_1| + \ldots + |C_{n-r_{t+1}}| + |R_{t+1}| &\leq r_{t+1} + \ldots + r_m \\
   &= (n-1) - (r_1 + \ldots + r_t) \\
   &\leq (n-1) - 2t,
   \end{aligned}$$

using (i) and the fact that $r_1 + \ldots + r_t + r_{t+1} + \ldots + r_m$ represents the total number of entries in our original PLS Q, which was equal to $n-1$.

Thus, in particular,

$$|C_{i_1}| + \ldots + |C_{i_k}| \leq (n-1) - 2t - |R_{t+1}| = (n-1) - 2t - r_{t+1}.$$

So, since sets $C_{i_2}, \ldots, C_{i_k}$ are non-empty, it follows that $|C_{i_1}| \leq (n-1) - 2t - r_{t+1} - (k-1) = n - 2t - r_{t+1} - k$.

Finally, we can deduce:

$$\begin{array}{rcl} |S_{i_1}| & \geq & |S| - |C_{i_1}| - |T_{i-1}| - |R_{t+1}| \\ & = & n - |C_{i_1}| - t - r_{t+1} \\ & \geq & n - (n - 2t - r_{t+1} - k) - t - r_{t+1} \\ & = & t + k \\ & \geq & k. \end{array}$$

So $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$.

2. Now we must assume that $k > n - t - r_{t+1}$. In particular, let $k = n - t - r_{t+1} + p$, where $p > 0$. As in case (1b) above, it can be shown that

$$|C_1| + \ldots + |C_{n-r_{t+1}}| + |R_{t+1}| \leq (n-1) - 2t.$$

So, since $|R_{t+1}| = r_{t+1}$, we have

$$|C_1| + \ldots + |C_{n-r_{t+1}}| \leq (n-1) - 2t - r_{t+1}.$$

Thus at least

$$(n - r_{t+1}) - [(n-1) - 2t - r_{t+1}] = 2t + 1$$

of the sets $C_1, \ldots, C_{n-r_{t+1}}$ must be empty. In particular, this implies that at least

$$\begin{array}{rcl} (2t+1) - [(n - r_{t+1}) - k] & = & (2t+1) - [(n - r_{t+1}) - (n - t - r_{t+1} + p)] \\ & = & (2t+1) - (t - p) \\ & = & t + p + 1 \\ & \geq & t + 1 \end{array}$$

of the sets $C_{i_1}, \ldots, C_{i_k}$ must be empty. So, without loss of generality, let the first $(t+1)$ of these empty sets be $C_{i_1}, \ldots, C_{i_{t+1}}$.

Now, for each $i = 1, \ldots, n - r_{t+1}$, let

$$S'_i = S \backslash S_i = C_i \cup T_i \cup R_{t+1}.$$

Note in particular that, since $C_{i_1}, \ldots, C_{i_{t+1}}$ are empty, we have $S'_i = T_i \cup R_{t+1}$ for $i = i_1, \ldots, i_{t+1}$.

We shall prove that

$$S'_{i_1} \cap \ldots \cap S'_{i_{t+1}} \subseteq R_{t+1}. \qquad \text{(iii)}$$

Choose any $x \in S'_{i_1} \cap \ldots \cap S'_{i_{t+1}}$. This means that $x \in T_i \cup R_{t+1}$, for each $i = i_1, \ldots, i_{t+1}$. If $x \notin R_{t+1}$, then it follows $x \in T_i$, for $i = i_1, \ldots, i_{t+1}$. Hence value x occurs at least $(t+1)$ times in the first t rows of Q_t.

However, this is impossible, since x occurs exactly once in each of the first t rows of Q_t and thus occurs exactly t times overall amongst these first t rows. So $x \notin R_{t+1}$ is impossible. Thus, for all $x \in S'_{i_1} \cap \ldots \cap S'_{i_{t+1}}$, we must have $x \in R_{t+1}$. Equation (iii) then follows.

Furthermore, since columns $i_1, \ldots, i_{t+1}$ belong to the collection $i_1, \ldots, i_k$, it follows that

$$S'_{i_1} \cap \ldots \cap S'_{i_k} \subseteq S'_{i_1} \cap \ldots \cap S'_{i_{t+1}} \subseteq R_{t+1},$$

using (iii) above. However, since $S_i = C_i \cup T_i \cup R_{t+1}$ for all i, we have

$$R_{t+1} \subseteq S'_{i_1} \cap \ldots \cap S'_{i_k}.$$

Thus

$$S'_{i_1} \cap \ldots \cap S'_{i_k} = R_{t+1}.$$

So, finally, we obtain

$$\begin{aligned} S_{i_1} \cup \ldots \cup S_{i_k} &= (S \backslash S'_{i_1}) \cup \ldots \cup (S \backslash S'_{i_k}) \\ &= S \backslash (S'_{i_1} \cap \ldots \cap S'_{i_k}) \\ &= S \backslash R_{t+1}, \end{aligned}$$

and hence

$$|S_{i_1} \cup \ldots \cup S_{i_k}| = n - r_{t+1} \geq k.$$

So, in all cases, we have $|S_{i_1} \cup \ldots \cup S_{i_k}| \geq k$. Thus, by Hall's Theorem, a SDR exists for sets $S_1, \ldots, S_{n-r_{t+1}}$.

Thus the $(t+1)$th row of Q_t can be filled as described earlier, producing an extension Q_{t+1} of Q.

Concluding Argument:

From the above reasoning, it can be seen that we can fill rows $1, \ldots, m$ of Q, one at a time, to obtain an extension Q_m of Q. Since each original entry of Q appears in the first m rows, it follows that the remaining rows $m+1, \ldots, n$ are empty.

Thus Q_m is an $m \times n$ latin rectangle. So Theorem 2.1.3 implies that there exists a completion L' of Q_m. Since Q_m is an extension of Q, we thus have a completion L' of Q.

Finally, permuting the rows using permutation α^{-1} will produce a latin square L that is a completion of P. Hence P is valid, as required.

If P contains less than $(n-1)$ entries:

Recall that, at the beginning of our proof, we required P to contain exactly $(n-1)$ entries. So now say P has less than $(n-1)$ entries. In particular, say P contains exactly $(n-1)-q$ entries, where $q > 0$.

Then there are at least q values that do not appear in P (in fact, there are at least $q+1$). Let these be $x_1, \ldots, x_q$. Furthermore, at least q columns of P are unused. Let these be $c_1, \ldots, c_q$.

Let r be some row already containing a value (if there is no such r, then P is an empty PLS, and so trivially contains a completion). Then cells (r, c_1), …, (r, c_q) are empty. Place value x_i in cell (r, c_i), for $i = 1, \ldots, q$. Let the resulting PLS be P'. Note that, since x_i does not already appear in P, neither the row latin nor the column latin condition is violated. Furthermore, since row r originally contained entries in P, the entries in P' are still contained within at most $(n/2)$ rows. Finally, note that P contains exactly $(n-1)$ entries. Thus, using our earlier argument, there is some completion L of P'.

However, since P' is an extension of P, it follows that L is a completion of P, as required.

Hence P is again valid, as required.

□

Applying the principle of symmetry then gives us two immediate corollaries of Theorem 3.2.1.

Corollary 3.2.2. *Let P be a PLS of order n, containing at most $n-1$ entries. If these entries lie in at most $n/2$ of the columns of P, then P is valid.*

Corollary 3.2.3. *Let P be a PLS of order n, containing at most $n-1$ entries. If at most $n/2$ different values appear in P, then P is valid.*

3.3 Back Diagonal Constructions

After providing the necessary definitions, we will present a series of constructions that, when combined, form the basis for Smetaniuk's proof of Evans' Conjecture.

Definition 3.3.1. Let P be a PLS of order n. Then the *back diagonal* of P is formed by cells $(1, n)$, $(2, n-1)$, …, $(n, 1)$.

Example 3.3.2. In the following example, the back diagonal of an empty PLS is marked with asterisks.

				*
			*	
		*		
	*			
*				

Definitions 3.3.3. Let P be a PLS of order n. Note that cell (x, y) lies on the back diagonal of P if and only if $x + y = n + 1$. If $x + y < n + 1$, we say cell (x, y) lies *above* the back diagonal of P. If $x + y > n + 1$, we say cell (x, y) lies *below* the back diagonal of P.

Example 3.3.4. In the following example, the cells above the back diagonal of an empty PLS are marked with a plus (+), and the cells below the back diagonal are marked with a minus (−).

+	+	+	+	
+	+	+		−
+	+		−	−
+		−	−	−
	−	−	−	−

Our first construction is then as follows.

Construction 3.3.5. Let L be a latin square of order n. Then $P(L)$ is the PLS of order $n+1$ formed as follows:

Choose any cell (x, y) in $P(L)$. Then the contents of cell (x, y) are determined by the following rules:

- If cell (x, y) lies on the back diagonal of $P(L)$, it contains the value $(n + 1)$.
- If cell (x, y) lies above the back diagonal of $P(L)$, it contains the corresponding value in cell (x, y) of L.
- If cell (x, y) lies below the back diagonal of $P(L)$, it remains empty.

Example 3.3.6. In the following example, a latin square L of order 8 and its corresponding PLS $P(L)$ are shown. This example, taken from [2], will be used throughout this section.

$$L = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 8 & 7 & 2 & 3 & 4 & 5 & 6 \\ \hline 6 & 4 & 1 & 8 & 5 & 3 & 7 & 2 \\ \hline 5 & 3 & 6 & 1 & 7 & 8 & 2 & 4 \\ \hline 3 & 5 & 8 & 4 & 2 & 6 & 1 & 5 \\ \hline 4 & 2 & 3 & 6 & 1 & 5 & 8 & 7 \\ \hline 2 & 6 & 5 & 3 & 8 & 7 & 4 & 1 \\ \hline 8 & 5 & 2 & 7 & 4 & 1 & 6 & 3 \\ \hline 7 & 1 & 4 & 5 & 6 & 2 & 3 & 8 \\ \hline \end{array}\,, \quad P(L) = \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 1 & 8 & 7 & 2 & 3 & 4 & 5 & 6 & 9 \\ \hline 6 & 4 & 1 & 8 & 5 & 3 & 7 & 9 & \\ \hline 5 & 3 & 6 & 1 & 7 & 8 & 9 & & \\ \hline 3 & 5 & 8 & 4 & 2 & 9 & & & \\ \hline 4 & 2 & 3 & 6 & 9 & & & & \\ \hline 2 & 6 & 5 & 9 & & & & & \\ \hline 8 & 5 & 9 & & & & & & \\ \hline 7 & 9 & & & & & & & \\ \hline 9 & & & & & & & & \\ \hline \end{array}\,.$$

Lemma 3.3.7. *$P(L)$ does in fact form a PLS, i.e. both the row latin and column latin conditions are satisfied.*

Proof. Any two entries in the same row must be of one of the following forms:

- Both are entries from L, in which case they cannot take the same value, since L is a latin square;
- One is an entry from L and the other lies on the back diagonal, in which case they cannot take the same value, since the back diagonal contains only values $(n+1)$ and L contains only values from $\{1, \ldots, n\}$.

Thus the row latin condition is satisfied. By a similar argument, the column latin condition is also satisfied. $\square$

Theorem 3.3.8. *Let L be any latin square. Then the PLS $P(L)$ is valid.*

In order to prove this, we will present a direct construction of Smetaniuk's that produces a completion of $P(L)$. The construction itself is based upon that presented in [2]. However, all the proofs of its properties and its correctness are my own.

Construction 3.3.9. The construction will be performed by completing one column of $P(L)$ at a time.

In particular, let M be an extension of $P(L)$ obtained by completing columns $1, \ldots, k$ of $P(L)$. For $i = 1, \ldots, n$, let M_i denote the set of values occurring in cells $(i, 1)$, …, (i, k) of M. Similarly, let L_i denote the set of values occurring in cells $(i, 1)$, …, (i, k) of L.

Then we define M to be a *cunning extension* of $P(L)$ if the following condition is satisfied:

- For each $i = n - k + 2, n - k + 3, \ldots, n$, we have $M_i \backslash \{n + 1\} \subseteq L_i$.

For instance, consider Example 3.3.6. Let M be the following extension obtained by completing columns $1, \ldots, 6$ (so in this case, $k = 6$).

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

Then, in order for M to be a cunning extension, the condition $M_i \backslash \{n+1\} = M_i \backslash \{9\} \subseteq L_i$ must be satisfied for $i = n-k+2, n-k+3, \ldots, n$. Using $n = 8$ and $k = 6$, this range for i becomes $i = 4, 5, \ldots, 8$. The corresponding sets M_i and L_i are then:

$$\begin{aligned} M_4 &= \{3, 5, 8, 4, 2, 9\}, & L_4 &= \{3, 5, 8, 4, 2, 6\}, \\ M_5 &= \{4, 2, 3, 6, 9, 1\}, & L_5 &= \{4, 2, 3, 6, 1, 5\}, \\ M_6 &= \{2, 6, 5, 9, 8, 7\}, & L_6 &= \{2, 6, 5, 3, 8, 7\}, \\ M_7 &= \{8, 5, 9, 7, 4, 2\}, & L_7 &= \{8, 5, 2, 7, 4, 1\}, \\ M_8 &= \{7, 9, 4, 5, 1, 6\}, & L_8 &= \{7, 1, 4, 5, 6, 2\}. \end{aligned}$$

In all five cases above, we see that the condition $M_i \backslash \{9\} \subseteq L_i$ is in fact satisfied. So M, as shown above, is indeed a cunning extension of $P(L)$.

Inductive argument:

The procedure for completing $P(L)$ will then be as follows:

1. Begin with $P(L)$. This is itself a cunning extension of $P(L)$ with $k = 1$ column(s) completed. This can be seen as follows.

   Since $k = 1$, we find that $n - k + 2 = n + 1$. So the range $i = n - k + 2, \ldots, n$, for which we require $M_i \backslash \{n+1\} \subseteq L_i$, is empty. Thus the "cunningness" condition is trivially satisfied.

2. Now assume we have a cunning extension M of $P(L)$ that has k columns completed. We then use the procedure described below to fill the $(k+1)$th column, so as to produce a cunning extension M' of $P(L)$ that has $(k+1)$ columns completed.

3. Step 2 can be repeated inductively until we have produced a cunning extension M^* of $P(L)$ in which n columns are complete. Thus only the final column, column $(n+1)$, may contain empty cells.

4. Remove any entries in column $(n+1)$, thus creating a PLS N in which the first n columns are completely filled and the remaining column is empty. We may then appeal to Corollary 2.1.5 to form a completion N^* of N.

   We then propose that N^* is in fact a completion of M^*. Let $(i, n+1)$ be a cell removed from M^* in order to produce N, and say this cell contained the value x. Then, since the first n columns of M^* are complete, it follows that every possible value except for x occurs in the first n cells of row i. So, when N is completed to form N^*, the only possible value that may be placed in cell $(i, n+1)$ is x.

Thus any entry removed from M^* when forming N is replaced when N^* is formed. So N^* is indeed a completion of M^*.

5. Now, since M^* is an extension of $P(L)$, and N^* is a completion of M^*, it thus follows that N^* is a completion of $P(L)$, as required.

Completing a column:

We now give the procedure referred to in step 2 above. Let M be a cunning extension of $P(L)$ in which columns $1, \ldots, k$ are complete, where $1 \leq k \leq n-1$. It is then our task to fill column $(k+1)$, in such a fashion that the resulting PLS is also a cunning extension of $P(L)$.

Row sequences:

For each value $x \in \{1, \ldots, n\}$, we construct the *row sequence* $\sigma(x) = \langle r_1, r_2, \ldots \rangle$ as follows.

1. If x does not appear in row $(n+1)$ of M, we simply define $\sigma(x) = \langle \rangle$, and our row sequence is complete.

   Otherwise, let c_1 be the column for which x appears in cell $(n+1, c_1)$ of M. Then r_1 is the row for which x appears in cell (r_1, c_1) of L, and we move on to step 2.

   Note that, in the latter case, $1 \leq c_1 \leq n$, since column $(n+1)$ of $P(L)$ only contains the value $(n+1)$. Thus x appears somewhere within column c_1 of L, since L is a latin square, and so r_1 does exist.

2. Say we have defined $r_1, \ldots, r_i$. If x does not appear in row r_i of M, we define $\sigma(x) = \langle r_1, \ldots, r_i \rangle$, and our row sequence is complete.

   Otherwise, let c_{i+1} be the column for which x appears in cell (r_i, c_{i+1}) of M. Then r_{i+1} is the row for which x appears in cell (r_{i+1}, c_{i+1}) of L, and we repeat step 2 using a new value of i.

   As before, note that $1 \leq c_{i+1} \leq n$, and so x appears within column c_{i+1} of L. Hence r_{i+1} exists.

Let us continue our example. We will evaluate $\sigma(1)$. To begin, note that value 1 appears in row $(n+1) = 9$ of M, and this is in cell $(9, 2)$. Thus $c_1 = 2$. Furthermore, 1 appears in cell $(8, 2)$ of L, and so $r_1 = 8$.

Value 1 appears in row 8 of M, and this is in cell $(8, 5)$. So $c_2 = 5$. Furthermore, 1 appears in cell $(5, 5)$ of L, so $r_2 = 5$. Next, we see that 1 appears in row 5 of M, in cell $(5, 6)$. So $c_3 = 6$. Then 1 lies in cell $(7, 6)$ of L, and thus $r_3 = 7$.

Finally, value 1 does not appear in row 7 of M. Hence our sequence is complete, and we have $\sigma(1) = \langle 8, 5, 7 \rangle$.

A full list of row sequences for our example is given below.

$$\begin{aligned}
\sigma(1) &= \langle 8, 5, 7 \rangle, \\
\sigma(2) &= \langle 7, 8 \rangle, \\
\sigma(3) &= \langle 6 \rangle,
\end{aligned}$$

$$\begin{aligned}
\sigma(4) &= \langle\rangle, \\
\sigma(5) &= \langle 5\rangle, \\
\sigma(6) &= \langle 8, 4\rangle, \\
\sigma(7) &= \langle\rangle, \\
\sigma(8) &= \langle\rangle.
\end{aligned}$$

Properties of row sequences:

We will now prove properties of the row sequences that will be of use later.

1. *If $\sigma(x) = \langle r_1, r_2, \ldots\rangle$, then $1 \leq r_i \leq n$, for all i:*

   This follows from the definition of $\sigma(x)$, since each r_i is a row of L.

2. *Members of $\sigma(x)$ are distinct, for all x:*

   Let $x \in \{1, \ldots, n\}$, and say $\sigma(x) = \langle r_1, r_2, \ldots\rangle$. Let $c_1, c_2, \ldots$ be as in the definition of $\sigma(x)$.

   Furthermore, say the members of $\sigma(x)$ are not all distinct. Then there must be some i, j, $i < j$, for which $r_i = r_j$. Choose the smallest such i.

   Say $i > 1$. Then, since x occurs in cells (r_i, c_i) and (r_j, c_j) of L and since $r_i = r_j$, it follows from the row latin condition of L that $c_i = c_j$. This in turn means that cells (r_{i-1}, c_i) and (r_{j-1}, c_j) of M both contain x. Since $c_i = c_j$, the column latin condition of M then tells us that $r_{i-1} = r_{j-1}$. However, this contradicts our choice of minimum i.

   So we must have $i = 1$. Again, x occurs in cells (r_i, c_i) and (r_j, c_j) of L, and so $c_i = c_j$. However, since $i = 1$, we now have x belonging to cells $(n+1, c_i)$ and (r_{j-1}, c_j) of M. Since $c_i = c_j$, the column latin condition of M implies that $r_{j-1} = n+1$. This is impossible, since $r_{j-1} \in \{1, \ldots, n\}$, by definition of the row sequence. So, in all cases, a contradiction has arisen.

   Thus the members of $\sigma(x)$ must be distinct, as required.

3. *$\sigma(x)$ is a finite sequence, for all x:*

   Say there is some x for which $\sigma(x)$ is infinite. Let $\sigma(x) = \langle r_1, r_2, \ldots\rangle$. Then, since r_i belongs to the finite set $\{1, \ldots, n\}$ for all i, we must have two members of $\sigma(x)$ equal. But this contradicts property 2 above.

   Thus $\sigma(x)$ cannot be infinite.

4. *If $\sigma(x) = \langle r_1, r_2, \ldots\rangle$ and $c_1, c_2, \ldots$ are as in the definition of $\sigma(x)$, then cell (r_i, c_i) lies below the back diagonal of L, for each i.*

   We shall prove this by induction on i. Notice first that every cell (r, c) not below the back diagonal of L contains the same value as the cell (r, c) of M. This is by definition of $P(L)$, and from the fact that M is an extension of $P(L)$.

   Say $\sigma(x) = \langle r_1, \ldots, r_t\rangle$. If $t = 0$, there is nothing to prove. So we shall assume $t \geq 1$.

   First, consider $i = 1$. By definition of r_1, we know that cell (r_1, c_1) of L contains value x. Assume that (r_1, c_1) is not below the back diagonal of L. Then cell (r_1, c_1) of M also contains value x. But, again by definition of r_1, we know cell $(n+1, c_1)$ of M contains value x. So the column latin property of M implies that $r_1 = n+1$, which is impossible, since it was proven earlier that $1 \leq r_i \leq n$, for all i. Thus cell (r_1, c_1) lies below the back diagonal of L.

So now assume that $1 < i \leq t$, and that cell (r_i, t_i) does not lie below the back diagonal of L. By definition of r_i, we know that cell (r_i, c_i) of L contains value x. Thus cell (r_i, c_i) of M also contains value x. However, again by definition of r_i, we know cell (r_{i-1}, c_i) of M contains value x. So the column latin property of M implies that $r_i = r_{i-1}$, which contradicts property 2 above. Hence cell (r_i, c_i) lies below the back diagonal of L.

Thus, by induction, cell (r_i, c_i) lies below the back diagonal of L, for all i.

5. *If $\sigma(x) = \langle r_1, r_2, \ldots \rangle$ and $c_1, c_2, \ldots$ are as in the definition of $\sigma(x)$, then $1 \leq c_i \leq k$, for each i:*

   Say $c_i > k$, for some i. If $i = 1$, then cell $(n+1, c_1)$ of M contains value x. Thus, since only cells $(n+1, 1)$, …, $(n+1, k)$ in row $(n+1)$ of M have been filled, it follows that $c_1 \leq k$, a contradiction.

   So say $i > 1$. Then cell (r_{i-1}, c_1) of M contains value x. Note that the only cells of M in columns $k+1, k+2, \ldots, n$ that are filled lie on or above the back diagonal of M. So, since $c_1 > k$, it follows that cell (r_{i-1}, c_1) of M lies on or above the back diagonal.

   All cells lying on the back diagonal of M contain value $(n+1)$. So, since cell (r_{i-1}, c_i) contains value $x \leq n$, this cell must lie strictly above the back diagonal. Hence, by construction of $P(L)$, cell (r_{i-1}, c_i) must also contain value x in L.

   So, since cell (r_i, c_i) of L also contains value x, the column latin property of L implies that $r_{i-1} = r_i$. But this contradicts property 2 above.

   Thus the required property of rows sequences is true.

Starting rows:

Now, for each value $x \in \{1, \ldots, n\}$, we define the *starting row* $r(x)$ as follows:

- If $\sigma(x) = \langle \rangle$, we have $r(x) = n + 1$.
- Otherwise, $r(x)$ is the last member of $\sigma(x)$. That is, if $\sigma(x) = \langle r_1, \ldots, r_t \rangle$, we have $r(x) = r_t$.

A full list of starting rows for our example is provided below.

$$\begin{aligned} r(1) &= 7, \\ r(2) &= 8, \\ r(3) &= 6, \\ r(4) &= 9, \\ r(5) &= 5, \\ r(6) &= 4, \\ r(7) &= 9, \\ r(8) &= 9. \end{aligned}$$

Properties of starting rows:

We again prove properties that will be required further into the construction.

1. *$r(x) \in \{1, \ldots, n\}$ if and only if $\sigma(x) \neq \langle\rangle$:*

   This is immediate from the definition of $r(x)$. If $\sigma(x) = \langle\rangle$, then $r(x) = n + 1$.

   Alternatively, let $\sigma(x) = \langle r_1, \ldots, r_t\rangle$. Then, from the properties of row sequences, $1 \leq r_t \leq n$. So, since $r(x) = r_t$, we have $1 \leq r(x) \leq n$, as required.

2. *For all x, value x does not belong to row $r(x)$ of M:*

   This again follows immediately from the definitions of $r(x)$ and $\sigma(x)$. If $r(x) = n+1$, then property 1 above tells us $\sigma(x) = \langle\rangle$. So, by definition of $\sigma(x)$, x does not appear in row $n + 1 = r(x)$ of M, and the required property holds.

   Otherwise, let $\sigma(x) = \langle r_1, \ldots, r_t\rangle$. Then, again by definition of $\sigma(x)$, x does not appear in row $r_t = r(x)$ of M. So again the required property is true.

3. *$r(x) \geq n - k + 2$, for all x:*

   If $\sigma(x) = \langle\rangle$, then
   $$r(x) = n + 1 = n - 1 + 2 \geq n - k + 2,$$
   as required. Otherwise, let $\sigma(x) = \langle r_1, \ldots, r_t\rangle$, where $t \geq 1$, and let $c_1, \ldots, c_t$ be as in the definition of $\sigma(x)$. From property 1, we have $r(x) \leq n$.

   From property 2, we see that value x does not belong to row $r(x)$ of M. However, x must belong to row $r(x)$ of L, since $r(x) \leq n$. In particular, x belongs to cell $(r(x) = r_t, c_t)$ of L. So this cell must be below the back diagonal of L (since otherwise, its value would also appear in M).

   From property 5 of row sequences, we know that $1 \leq c_t \leq k$. So, since the cell $(r(x), c_t)$ lies below the back diagonal of L, it follows that $r(x) \geq n - k + 2$, as required.

4. *If $r(x) = r(y)$, then either $x = y$ or $r(x) = r(y) = n + 1$:*

   To the contrary, say $r(x) = r(y) = r_0 \in \{1, \ldots, n\}$ and that $x \neq y$.

   From property 1 above, we can assume that $\sigma(x) = \langle r_1, \ldots, r_p\rangle$ and that $\sigma(y) = \langle s_1, \ldots, s_q\rangle$. Let the corresponding columns be $c_1, \ldots, c_p$ and $d_1, \ldots, d_q$, as in the definitions of $\sigma(x)$ and $\sigma(y)$.

   From property 2 above, we see that neither x nor y belong to row r_0 of M. However, both x and y must belong to row r_0 of L. In particular, x belongs to cell $(r_0 = r_p, c_p)$ of L and y belongs to cell $(r_0 = s_q, d_q)$ of L.

   From property 3 above, we see that $r_0 \geq n - k + 2$. Now recall the definition of a cunning extension. We know that neither x nor y appear in row r_0 of M. Thus $x, y \notin M_{r_0}$. However, both x and y appear in the first k cells of row r_0 of L (since $c_p, d_q \leq k$, from property 5 of row sequences). Hence $x, y \in L_{r_0}$. Furthermore, since $r_0 \geq n - k + 2$, the fact that M is a cunning extension of $P(L)$ tells us that $M_{r_0} \backslash \{n + 1\} \subseteq L_{r_0}$.

   Now comes our contradiction. We know $|L_{r_0}| = k$. Furthermore, since $r_0 \geq n-k+2$, row r_0 of M contains exactly k entries, including one that takes value $(n+1)$. Thus $|M_{r_0} \backslash \{n+1\}| = k - 1$. So it follows that at most one member of L_{r_0} may be absent from M_{r_0}.

   Since both x, y belong to L_{r_0} and both are absent from M_{r_0}, it then follows that $x = y$.

Value sequence:

We will now construct the *value sequence* $\nu = \langle x_1, x_2, \ldots \rangle$ as follows.

1. Let x_1 be the value in cell $(n-k+1, k+1)$ of L.

2. Say we have defined $x_1, \ldots, x_i$. If $r(x_i) = n+1$, then our construction is complete. Let $\nu = \langle x_1, \ldots, x_i \rangle$.

   Otherwise, let x_{i+1} be the value in cell $(r(x_i), k+1)$ of L. Then repeat step 2, using a new value of i.

Again, we will consider our example. In this case, the value in cell $(n-k+1, k+1) = (8-6+1, 6+1) = (3,7)$ of L is 2. So $x_1 = 2$.

Continuing, the value in cell $(r(2), k+1) = (8,7)$ of L is 3. So $x_2 = 3$. Then the value in cell $(r(3), 7) = (6,7)$ of L is 4, so $x_3 = 4$.

Finally, $r(4) = 9$, and so our our construction terminates. We then have the completed value sequence

$$\nu = \langle 2, 3, 4 \rangle.$$

Properties of the value sequence:

Again, we will discuss properties of our new construct.

1. *Any two members of ν are distinct:*

   Say $\nu = \langle x_1, x_2, \ldots \rangle$. Furthermore, say $x_i = x_j$ for some i, j with $i < j$. Choose the smallest such i.

   Since we terminate construction of ν at the first appearance of value $(n+1)$, it follows that ν can contain value $(n+1)$ at most once. Thus $x_i, x_j \neq (n+1)$.

   Now say $i > 1$. Then x_i is the value in cell $(r(x_{i-1}), k+1)$ of L, and x_j is the value in cell $(r(x_{j-1}), k+1)$ of L. Since $x_i = x_j$, the column latin property of L thus implies $r(x_{i-1}) = r(x_{j-1})$. Thus property 4 of starting rows implies one of the following:

   - $r(x_{i-1}) = r(x_{j-1}) = n+1$:
     If this were true, then construction of ν would have terminated at x_{i-1}, i.e. $\nu = \langle x_1, \ldots, x_{i-1} \rangle$. Thus x_i and x_j cannot be members of ν.
   - $x_{i-1} = x_{j-1}$:
     This contradicts the minimality of i.

   Both cases lead to a contradiction. Hence $i = 1$.

   So x_1 is in cell $(n-k+1, k+1)$ of L. Furthermore, since $j > i = 1$, we know x_j is the value in cell $(r(x_{j-1}), k+1)$ of L. Thus, from the column latin condition of L, it follows that $r(x_{j-1}) = n-k+1$. But this is a contradiction of property 3 of starting rows.

   Hence any two members of ν must be distinct.

2. *If $\nu = \langle x_1, \ldots, x_m \rangle$, then the rows $r(x_1), \ldots, r(x_m)$ are distinct:*

   Say $r(x_i) = r(x_j)$, for some i, j with $i < j$. If $r(x_i) = r(x_j) \leq n$, then property 4 of starting rows implies $x_i = x_j$, a contradiction.

So $r(x_i) = r(x_j) = n + 1$. But then, since $i < j$, construction of the value sequence would have terminated at x_i (by definition of ν), and hence x_j would not be a member of ν.

Thus the required property of the value sequence is true.

Filling cells:

Finally, we are at a state at which we can fill column $(k+1)$ of M, producing our new extension M'. Let $\nu = (x_1, \ldots, x_m)$. The procedure is then as follows:

1. For $i = 1, \ldots, m$, fill cell $(r(x_i), k+1)$ of M with value x_i.
2. In the remaining rows r for which cell $(r, k+1)$ has not yet been filled, fill cell $(r, k+1)$ of M with the corresponding value in cell $(r, k+1)$ of L.

Again continuing our example, recall that $\nu = \langle 2, 3, 4 \rangle$, and that

$$\begin{aligned} r(2) &= 8, \\ r(3) &= 6, \\ r(4) &= 9. \end{aligned}$$

So filling cells $(8,7)$, $(6,7)$ and $(9,7)$ with values $2, 3, 4$ respectively gives the PLS shown below.

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

We then fill the remaing cells of column $k + 1 = 7$ with the corresponding entries in L, producing the new extension

$M' =$

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

.

Proof that construction is possible:

Here, we shall prove that no cell is “overwritten” by the above construction.

- First, note that the empty cells in column $(k+1)$ of M were precisely those of the form $(r, k+1)$ where $r \geq n-k+2$. In, step 1 of the above construction, only cells of the form $(r(x), k+1)$ were filled. From property 3 of starting rows, we know $r(x) \geq n-k+2$, for all x. So step 1 of our construction does not overwrite existing entries in M.
- Furthermore, property 2 of the value sequence implies that cells $(r(x_1), k+1)$, ..., $(r(x_m), k+1)$ are distinct. Thus no cell has been filled twice by step 1.
- Since step 2 fills only empty cells, it causes no cells to be overwritten.

Proof that column $(k+1)$ is filled:

Since step 2 fills every empty cell in column $(k+1)$ except for $(n+1, k+1)$, our only duty is prove that cell $(n+1, k+1)$ of M is filled by step 1. However, let $\nu = \langle x_1, \ldots, x_m \rangle$. Then, by definition of ν, $r(x_m) = n+1$. Hence step 1 places value x_m in cell $(n+1, k+1)$.

Proof that M' is a PLS:

We will split this into proof of the row latin and column latin conditions separately.

- *Proof of row latin condition:* First, consider step 1. This fills cells of the form $(r(x), k+1)$ with value x. From property 2 of starting rows, value x does not already appear in row $r(x)$ of M. Thus the row latin condition is not violated by step 1.

  We must also consider step 2. Let $(r, k+1)$ be a cell of M filled by this step, and let it be filled with value x. Then x appears in cell $(r, k+1)$ of L. So $x \notin L_r$, since x cannot belong to the first k entries of row r of L.

  Now recall that M was a cunning extension of $P(L)$. Thus $M_r \backslash \{n+1\} \subseteq L_r$. So, since $x \neq n+1$ and $x \notin L_r$, we see that $x \notin M_r$. Hence x does not already appear in row r of M, and so the row latin condition is not violated by step 2.

- *Proof of column latin condition:*

  From property 1 of the value sequence, we know the values placed in column $(k+1)$ by step 1 are distinct. Furthermore, from the column latin condition of L, we know the values placed in column $(k+1)$ by step 2 are also distinct.

  Let $\nu = \langle x_1, \ldots, x_m \rangle$. Then the following facts remain to be proven:

  - *The same value is not placed in column $(k+1)$ by both steps 1 and 2:*
    Say value x_i is placed in column $(k+1)$ by step 1. If $i = 1$, then cell $(n-k+1, k+1)$ of L contains value x_i (by definition of ν). But, in M, this cell belongs to the back diagonal and thus contains value $n+1$. Since this is the only occurrence of x_i in column $(k+1)$ of L, x_i is not placed in column $(k+1)$ by step 2.

    Say, on the other hand, that $i > 1$. Then cell $(r(x_{i-1}), k+1)$ of L contains value x_i (by definition of ν). However, step 1 places value x_{i-1} in cell $(r(x_{i-1}), k+1)$ of M. Since cell $(r(x_{i-1}), k+1)$ is the only occurrence of value x_i in column $(k+1)$ of L, we again see that x_i cannot be placed in column $(k+1)$ by step 2.

- *Step 1 does not place a value in column $(k+1)$ of M that existed in that column before construction:*

  To the contrary, say some value x_i is placed in column $(k+1)$ by step 1, where x_i existed in that column before construction. In particular, say x_i existed in cell $(r, k+1)$. Then this cell must lie above the back diagonal of M, and hence cell $(r, k+1)$ of L also contains value x_i (it cannot lie on the back diagonal of M, since the back diagonal only holds value $(n+1)$). Since this cell lies above the back diagonal of M, we have $r < n-k+1$.

  If $i = 1$, we again see that cell $(n-k+1, k+1)$ of L contains value x_i. This raises a contradiction, since cell $(r, k+1)$ also contains x_i, and we know $r < n-k+1$.

  So $i > 1$. Thus, as previously, we find that cell $(r(x_{i-1}), k+1)$ of L contains value x_i. However, we also have cell $(r, k+1)$ of L containing x_i, where $r < n-k+1$. Since property 3 of starting rows implies $r(x_{i-1}) \geq n-k+2$, a contradiction again arises.

- *Step 2 does not place a value in column $(k+1)$ of M that existed in that column before construction:*

  Step 2 only adds entries to M that already exist in L. As mentioned previously, the only entries existing in column $(k+1)$ of M before construction are:

  * The entries above the back diagonal, which already exist in L;
  * The entry on the back diagonal, whose value is $(n+1)$.

  Since two different entries from column $(k+1)$ of L cannot have the same value, no previously existing values above the back diagonal of M can have the same value as any entered during step 2. Furthermore, since no entries of L have value $(n+1)$, the previously existing value on the back diagonal of M cannot have the same value as any entered during step 2.

This then proves the column latin condition for M'.

Hence M' is a PLS, as required..

Proof that M' is cunning:

Finally, we must prove the "cunningness" of M', in order to allow the induction to continue.

For $i = 1, \ldots, n$, let M_i' denote the set of values occurring in cells $(i,1)$, …, $(i, k+1)$ of M'. Similarly, let L_i' denote the set of values occurring in cells $(i,1)$, …, $(i,k+1)$ of L. We must prove, for $i = n-(k+1)+2, n-(k+1)+3, \ldots, n$, that $M_i' \backslash \{n+1\} \subseteq L_i'$.

We will take two cases.

- If $i = n-(k+1)+2$:

  Then $i = n-k+1$, and the only entries of M' in row i are those on or above the back diagonal. Since the entry on the back diagonal contains value $(n+1)$ and those above the back diagonal contain the same values as cells $(i,1), \ldots, (i,k)$ of L, we see in this case that $M_i' \backslash \{n+1\} \subseteq L_i'$, as required.

- If $n-(k+1)+2 < i \leq n$:

  Then $i \geq n-k+2$, and so we can use the cunning property of M to deduce that $M_i \backslash \{n+1\} \subseteq L_i$. Let the value in cell $(i, k+1)$ of M' be x. Then $M_i' = M_i \cup \{x\}$.

If this cell was filled by step 2 of our construction, then cell $(i, k+1)$ of L also contains x. Thus $L'_i = L_i \cup \{x\}$, and it follows that $M'_i \backslash \{n+1\} \subseteq L'_i$, as required.

Otherwise, this cell was filled by step 1. Thus $i = r(x)$. Since $i \leq n$, property 1 of starting rows shows that $\sigma(x) = \langle r_1, \ldots, r_t \rangle$, where $t \geq 1$. Let the corresponding columns from the definition of $\sigma(x)$ be $c_1, \ldots, c_t$.

Then x appears in cell $(r_t = r(x), c_t)$ of L. Furthermore, property 5 of row sequences shows that $c_t \leq k$. Thus $x \in L_i$. In particular, this implies $x \in L'_i$. From this, we can deduce $M'_i \backslash \{n+1\} \subseteq L'_i$, as required.

Hence M' is a cunning extension of $P(L)$ with $(k+1)$ columns complete.

We can now finally present a proof of Theorem 3.3.8!

Proof. Construction 3.3.9 produces a completion of $P(L)$. Thus $P(L)$ is valid. $\square$

3.4 Permuting Rows and Columns

The next component of Smetaniuk's proof of Evans' Conjecture involves rearranging the rows and columns of particular classes of PLSs, in such a manner that certain desirable properties hold. In this section, the appropriate construction will be presented.

The following construction and the accompanying proof of correctness are expanded upon those presented in [5].

Lemma 3.4.1. *Let P be a PLS of order n, containing at most $n-1$ entries. Furthermore, say there is some value x appearing exactly once in P.*

Then the rows and columns of P can be rearranged so that:

- *The single entry with value x lies upon the back diagonal of P;*
- *The remaining entries appear above the back diagonal of P.*

Example 3.4.2. The PLS shown below satisfies the conditions for Lemma 3.4.1. We will choose $x = 4$ to represent the value appearing exactly once.

$$P = \begin{array}{|c|c|c|c|c|c|} \hline 1 & & & & & \\ \hline & 3 & & 5 & & \\ \hline & & & & & \\ \hline & 4 & & & & \\ \hline & & & 1 & & \\ \hline & & & & & \\ \hline \end{array} \, .$$

We will not present an appropriate rearrangement at this stage. Instead, this example will be referred to throughout Construction 3.4.3.

The proof of Lemma 3.4.1 requires Construction 3.4.3, which is presented below.

Construction 3.4.3. Move the row containing value x to the top of the PLS. Move any empty rows to the bottom. Now say P contains exactly m non-empty rows. Then label these non-empty rows with *symbols* $r_1, \ldots, r_m$, in order from top to bottom, so that value x appears in row r_1. Note that, when these rows are moved, their corresponding symbols will move with them.

So, in our example, we must move the row containg value 4 to the top of the PLS. P then becomes:

r_1		4				
r_2	1					
r_3		3		5		
r_4				1		

We shall let n_{r_i} denote the number of entries in row r_i, where $i \in \{1, \ldots, m\}$. For our example, the corresponding values are:

$$\begin{aligned} n_{r_1} &= 1; \\ n_{r_2} &= 1; \\ n_{r_3} &= 2; \\ n_{r_4} &= 1. \end{aligned}$$

For $k \in \{1, \ldots, m\}$, let $N(k)$ denote the total number of entries in rows $r_1, \ldots, r_k$. That is,

$$N(k) = \sum_{i=1}^{k} n_{r_i}.$$

Since rows $r_1, \ldots, r_m$ are non-empty, it follows that the totals $N(1), \ldots, N(m)$ are distinct and increasing. In our example, the corresponding sums are:

$$\begin{aligned} N(1) &= 1; \\ N(2) &= 2; \\ N(3) &= 4; \\ N(4) &= 5. \end{aligned}$$

Since row r_m is the last non-empty row, the total $N(m) = \sum_{i=1}^{m} n_{r_i}$ represents the total number of entries in P. Thus $N(m) \leq n - 1$. Furthermore, since row r_1 is non-empty, we have $N(1) = n_{r_1} \geq 1$. So, combined with the fact that $N(1), \ldots, N(m)$ are increasing, we have

$$1 \leq N(1) < \ldots < N(m) \leq n - 1. \tag{iv}$$

Now rearrange the rows so that, for $i = 1, \ldots, m$, row r_i becomes the $(n - N(i))$th row. Note that, since $N(1), \ldots, N(m)$ are distinct, no clashes will arise. Furthermore, (iv) above implies that $1 \leq n - N(i) \leq n - 1$ for $i = 1, \ldots, m$, and so the specified row positions do indeed exist.

Continuing with our example:

$$\begin{aligned} n - N(1) &= 5; \\ n - N(2) &= 4; \\ n - N(3) &= 2; \\ n - N(4) &= 1. \end{aligned}$$

So, for instance, row r_1 becomes the fifth row and row r_3 becomes the second row. The

PLS thus produced is shown below.

r_4				1		
r_3		3		5		
r_2	1					
r_1		4				

Rearranging columns:

The procedure for rearranging columns will now be described. We will define integers $c_1, \ldots, c_m$, satisfying $c_1 \leq \ldots \leq c_m$, in such a manner that the entries in row r_i appear in columns $1, \ldots, c_i$ (although not necessarily in all of these columns), for $i = 1, \ldots, m$. Furthermore, we will prove inductively that $c_i \leq N(i)$, again for $i = 1, \ldots, m$.

1. First, permute the columns so that the n_{r_1} entries in row r_1 appear in columns $1, \ldots, n_{r_1}$. Then let $c_1 = n_{r_1}$. Note then that, trivially, $c_1 \leq n_{r_1} = N(1)$.

   We will now declare columns $1, \ldots, c_{r_1}$ to be *fixed*, and these columns will not be moved again until further notice. Row r_1 is now declared to be *satisfied*.

   In our example, row r_1 contains the single value 4. So $c_1 = 1$. Our task is to move this single entry in row r_1 to the first column. To do this, we shall swap columns 1 and 2. Once this is done, column 1 will be fixed. Fixed columns will be marked with an asterisk ($*$).

	$*$					
r_4				1		
r_3	3			5		
r_2		1				
r_1	4					

2. Now assume we have satisfied rows $r_1, \ldots, r_i$, for some $i \in \{1, \ldots, m-1\}$, and that columns $1, \ldots, c_i$ are fixed. Assume also that $c_1 \leq \ldots \leq c_i$, and that $c_i \leq N(i)$.

   Consider row r_{i+1}, which contains $n_{r_{i+1}}$ entries. Say t of these entries do not belong to fixed columns. Then move these t columns to the left until they are in positions $c_i + 1, \ldots, c_i + t$ (i.e. adjacent to the fixed columns). Note that this is possible, since $c_i + t \leq N(i) + n_{r_{i+1}} = N(i+1) \leq n - 1$.

   Declare columns $c_i + 1, \ldots, c_i + t$ to be fixed. We will now define $c_{i+1} = c_i + t$.

   Note the following points.

   - The fixed columns are now $1, \ldots, c_{i+1}$.
   - Since $t \geq 0$, we have $c_i \leq c_{i+1}$. So $c_1 \leq \ldots \leq c_i \leq c_{i+1}$.
   - Since $t \leq n_{r_{i+1}}$, we have

     $$c_{i+1} = c_i + t \leq N(i) + n_{r_{i+1}} = N(i+1).$$

   - The entries in row r_i all appear in columns $1, \ldots, c_{i+1}$ (although not necessarily in all of these columns).

We now declare row r_{i+1} to be satisfied.

3. Step 2 is performed repeatedly until all rows $r_1, \ldots, r_m$ are satisfied.

Consider again our example. Since row r_1 has already been satisfied, we must now consider row r_2. This contains the single value 1, which does not appear in a fixed column. Thus $t = 1$, and so $c_2 = c_1 + t = 1 + 1 = 2$. Since the column containing this value 1 is already adjacent to the fixed columns, no columns need to be moved. Fix this column. Row r_2 is now satisfied, and the resulting PLS is shown below.

	*	*				
r_4				1		
r_3	3			5		
r_2		1				
r_1	4					

Now consider row r_3. This contains two values, namely 3 and 5. The entry with value 3 belongs to a fixed column, but the entry with value 5 does not. So $t = 1$, and $c_3 = c_2 + t = 2 + 1 = 3$. We must move the column containing this value 5 to a position adjacent to the fixed columns. This column is then fixed itself. Row r_3 has thus been satisfied, and the resulting PLS is shown below.

	*	*	*			
r_4			1			
r_3	3		5			
r_2		1				
r_1	4					

Finally, every entry in row r_4 belongs to a fixed column. Thus $t = 0$, $c_4 = c_3 + t = 3 + 0 = 3$, and no columns need to be either moved or fixed. Row r_4 is then satisfied, and the final PLS is identical to the previous PLS above.

In general, the following properties follow by induction:

- The fixed columns are $1, \ldots, c_m$.
- $c_1 \leq \ldots \leq c_m$.
- $c_i \leq N(i)$ for $i = 1, \ldots, m$.
- The entries in row r_i all appear in columns $1, \ldots, c_i$ (although not necessarily in all of these columns), for $i = 1, \ldots, m$.

The last two of these properties imply that, for each non-empty row r_i, the entries in row r_i appear within columns $1, \ldots, N(i)$, although again not necessarily in all of these columns. However, recall that row r_i is the $(n - N(i))$th row. So each entry in row i is in cell (r, c), where $r = n - N(i)$ and $c \leq N(i)$. Thus $r + c \leq n - N(i) + N(i) = n$, and so cell (r, c) lies above the back diagonal of the PLS (since back diagonal cells (r', c') satisfy $r' + c' = n + 1$).

Hence every entry of the resulting PLS lies above the back diagonal.

Moving x onto the back diagonal:

Now we shall describe the procedure for placing the value x on the back diagonal.

Recall that value x occurs in row r_1, which is the $(n - N(1))$th row of the PLS. Let this be in cell $(n - N(1), c)$. Recall also that each entry in row r_1 appears in one of the columns $1, \ldots, N(1)$. So $c \leq N(1)$.

Furthermore, recall that $1 \leq N(1) \leq n - 1$. So column $N(1) + 1$ exists.

Then the final step of our construction is to swap columns c and $N(1) + 1$. Note that this may require columns to be "unfixed"; we will now allow this.

In our example, the single value 4 appears in cell $(5, 1)$. So $c = 1$. Furthermore, we have $N(1) = 1$. Thus columns $c = 1$ and $N(1) + 1 = 2$ are to be swapped. This results in the following PLS:

$$Q = \begin{array}{|c|c|c|c|c|c|} \hline & & 1 & & & \\ \hline & 3 & 5 & & & \\ \hline & & & & & \\ \hline 1 & & & & & \\ \hline & 4 & & & & \\ \hline & & & & & \\ \hline \end{array} .$$

Note the following points.

- Consider any entry in column $N(1) + 1$, before the swap takes place. Let this be in cell $(r, N(1) + 1)$. Since we know this entry lies above the back diagonal, we have $r + N(1) + 1 \leq n$.

  The new location for this entry is (r, c). Then $r + c \leq r + N(1) < r + N(1) + 1 \leq n$. Thus, in its new location, the entry still lies above the back diagonal.

- Now consider any entry in column c, before the swap takes place, excluding the entry with value x that appears in row r_1. This entry must then belong to row r_i, for some $i \in \{2, \ldots, m\}$ (it cannot lie in row r_1, since value x already occupies that cell). So it is located in cell $(n - N(i), c)$. Its new location is then cell $(n - N(i), N(1) + 1)$. Furthermore, we have

  $$[n - N(i)] + [N(1) + 1] < n - N(1) + N(1) + 1 = n + 1,$$

  using the fact that $N(1) < N(i)$, as seen in equation (iv). So $[n-N(i)]+[N(1)+1] \leq n$. Thus, in its new location, the entry still lies above the back diagonal.

  Finally, consider the single entry with value x. Before the swap, this occupies cell $(n-N(1), c)$. So, after the swap, its location is cell $(n-N(1), N(1)+1)$. In particular, this gives $[n - N(1)] + [N(1) + 1] = n + 1$. So, in its new location, this entry lies exactly upon the back diagonal.

Hence, after the column swap is performed, the following conditions hold:

- The single entry with value x lies upon the back diagonal of P;
- The remaining entries appear above the back diagonal of P.

For instance, these properties can be seen in Q, which was shown earlier in our example.

We can now prove Lemma 3.4.1.

Proof. A rearrangement of rows and columns satisfying the required conditions is given in Construction 3.4.3. □

3.5 Smetaniuk's Proof Completed

At this stage, we can finally piece together the various results obtained in this chapter. The results will form Smetaniuk's proof of Evans' Conjecture, which will now be restated as a theorem.

Theorem 3.5.1 (Evans' Conjecture)**.** *Let P be a PLS of order n. If P contains at most $n-1$ entries, then P is valid.*

The following proof is based upon that given in [2].

Proof. Our proof will proceed by induction on n.

To begin with, note that Evans' Conjecture is trivially true for $n = 1$, since any PLS of order 1 containing at most 0 entries can be completed, as shown below.

$$\begin{array}{|c|}\hline 1 \\ \hline \end{array}$$

Furthermore, Evans' Conjecture holds for $n = 2$. Say we have a PLS P of order 2 containing at most 1 entry. Then, without loss of generality, we can assume P is one of the following PLSs:

$$\begin{array}{|c|c|}\hline & \\ \hline & \\ \hline \end{array}\,, \quad \begin{array}{|c|c|}\hline 1 & \\ \hline & \\ \hline \end{array}\,.$$

However, both of the above PLSs have the completion shown below.

$$\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array}$$

Thus P is valid, as required.

So now assume Evans' Conjecture holds for $n = k-1$, where $k > 2$. We must prove it true for $n = k$.

Let P be a PLS of order k, containing at most $k-1$ entries. If at most $k/2$ distinct values appear in P, then Corollary 3.2.3 implies that P is valid, as required.

Otherwise, P contains more than $k/2$ distinct values. Thus, since P contains at most $k - 1 < 2 \cdot (k/2)$ entries, there must be some value that appears exactly once. Let this value be x.

We shall now relabel the values of P so that x is relabelled to k. Such a relabelling can be represented as a permutation δ of $\{1, \dots, k\}$. Let the resulting PLS be P'. So the value k appears exactly once in P'.

Then, from Lemma 3.4.1, we can rearrange the rows and columns of P', using permutations α and β respectively, to obtain a PLS Q for which:

- The single entry with value k lies upon the back diagonal of Q;
- The remaining entries appear above the back diagonal of Q.

Now let Q' be the PLS obtained from Q by removing the single entry with value k. Thus all entries in Q' lie above the back diagonal of Q'. In particular, this implies that row k and column k of Q' are empty. So let R be the PLS of order $k-1$ obtained by removing row k and column k from Q'.

Then the entries in R are exactly the same as the entries in Q, except that the single entry of value k is absent from R. Furthermore, all entries in R occur on or above the back diagonal of R.

So, since Q is a PLS of order k containing at most $k-1$ entries, it follows that R is a PLS of order $k-1$ containing at most $k-2$ entries. Thus, using our inductive hypothesis, R is valid.

Then let L_0 be a completion of R, where L_0 is a latin square of order $n-1$. Consider $P(L_0)$, as defined in Construction 3.3.5. Since all entries in R occur on or above the back diagonal of R and since L_0 is a completion of R, it follows that all entries of R also occur in $P(L_0)$.

Furthermore, note that $P(L_0)$ is a PLS of order k, whose back diagonal is completely filled with entries of value k. Thus the entry with value k, removed from Q to form Q' and lying on the back diagonal of Q, also lies in $P(L_0)$.

Hence, since all other entries of Q are contained in R and hence in $P(L_0)$, it follows that all entries in Q are contained in $P(L_0)$. Furthermore, since Q and $P(L_0)$ are of the same order, it follows that $P(L_0)$ is an extension of Q.

Now, from Theorem 3.3.8, we see that $P(L_0)$ is valid. So let L' be a completion of $P(L_0)$. Then, since $P(L_0)$ is in turn an extension of Q, it follows that L' is a completion of Q.

Finally, note that Q is obtained from P by rearranging rows, columns and value labels using permutations α, β and δ respectively. So let L be the latin square obtained by rearranging the rows, columns and value labels of L' using permutations α^{-1}, β^{-1} and δ^{-1} respectively.

Then L is a completion of P, and so P is valid.

Theorem 3.5.1 hence follows by induction. □

Example 3.5.2. We will continue the example introduced in Construction 3.4.3. We begin with the PLS P, as show below. The number of distinct values appearing in P is $4 > (6/2)$, and so we cannot use Corollary 3.2.3 to complete P. Instead, we must proceed via Lemma 3.4.1 and the ensuing constructions.

We will choose $x = 4$ to represent a value that appears exactly once in P.

$$P = \begin{array}{|c|c|c|c|c|c|}\hline 1 & & & & & \\ \hline & 3 & & 5 & & \\ \hline & & & & & \\ \hline & 4 & & & & \\ \hline & & & 1 & & \\ \hline & & & & & \\ \hline \end{array}\,.$$

We now wish to relabel values so that 4 becomes 6. This is done simply by interchanging value labels 4 and 6. The resulting PLS is shown below.

$$P' = \begin{array}{|c|c|c|c|c|c|}\hline 1 & & & & & \\ \hline & 3 & & 5 & & \\ \hline & & & & & \\ \hline & 6 & & & & \\ \hline & & & 1 & & \\ \hline & & & & & \\ \hline \end{array}\,.$$

Our relabelling permutation is then $\delta = (4\ 6)$. Following this, the rearrangement of rows and columns, as illustrated in Construction 3.4.3 (but interchanging labels 4 and 6) gives

Q, as shown below.

$$Q = \begin{array}{|c|c|c|c|c|c|}\hline & & 1 & & & \\ \hline & 3 & 5 & & & \\ \hline & & & & & \\ \hline 1 & & & & & \\ \hline & 6 & & & & \\ \hline & & & & & \\ \hline \end{array}\,.$$

Rearrangement permutations are thus $\alpha = (1\ 4\ 5)$ for rows and $\beta = (3\ 4)$ for columns. We then remove the entry with value 6 to produce Q', as follows:

$$Q' = \begin{array}{|c|c|c|c|c|c|}\hline & & 1 & & & \\ \hline & 3 & 5 & & & \\ \hline & & & & & \\ \hline 1 & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}\,.$$

Following this, R is obtained by removing row 6 and column 6.

$$R = \begin{array}{|c|c|c|c|c|}\hline & & 1 & & \\ \hline & 3 & 5 & & \\ \hline & & & & \\ \hline 1 & & & & \\ \hline & & & & \\ \hline \end{array}\,.$$

We then find a completion L_0 of R.

$$L_0 = \begin{array}{|c|c|c|c|c|}\hline 3 & 4 & 1 & 2 & 5 \\ \hline 4 & 3 & 5 & 1 & 2 \\ \hline 5 & 1 & 2 & 4 & 3 \\ \hline 1 & 2 & 3 & 5 & 4 \\ \hline 2 & 5 & 4 & 3 & 1 \\ \hline \end{array}\,.$$

The next step is to produce $P(L_0)$.

$$P(L_0) = \begin{array}{|c|c|c|c|c|c|}\hline 3 & 4 & 1 & 2 & 5 & 6 \\ \hline 4 & 3 & 5 & 1 & 6 & \\ \hline 5 & 1 & 2 & 6 & & \\ \hline 1 & 2 & 6 & & & \\ \hline 2 & 6 & & & & \\ \hline 6 & & & & & \\ \hline \end{array}\,.$$

A completion L' of $P(L_0)$ is then obtained.

$$L' = \begin{array}{|c|c|c|c|c|c|}\hline 3 & 4 & 1 & 2 & 5 & 6 \\ \hline 4 & 3 & 5 & 1 & 6 & 2 \\ \hline 5 & 1 & 2 & 6 & 3 & 4 \\ \hline 1 & 2 & 6 & 3 & 4 & 5 \\ \hline 2 & 6 & 4 & 5 & 1 & 3 \\ \hline 6 & 5 & 3 & 4 & 2 & 1 \\ \hline \end{array}\,.$$

Note that L' is a completion of Q. Finally, we apply the inverses of permutations α, β and δ to the rows, columns and value labels of L' respectively, to obtain L. Note that $\alpha^{-1} = (1\ 5\ 4)$, $\beta^{-1} = (3\ 4)$ and $\delta^{-1} = (4\ 6)$.

$$L = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 6 & 5 \\ \hline 6 & 3 & 1 & 5 & 4 & 2 \\ \hline 5 & 1 & 4 & 2 & 3 & 6 \\ \hline 2 & 4 & 5 & 6 & 1 & 3 \\ \hline 3 & 6 & 2 & 1 & 5 & 4 \\ \hline 4 & 5 & 6 & 3 & 2 & 1 \\ \hline \end{array}\,.$$

Finally, note that L, as shown above, is indeed a completion of our original PLS P.

Chapter 4

Completing a k-Stagger

In this chapter, we examine an open problem upon which I have worked. We examine the class of k-staggers, which are PLSs in which each row, column and value is used in exactly k entries. In particular, we ask the following question:

For which values of k and n are all k-staggers of order n valid?

To begin with, a number of specific results are obtained, some of which are based upon computational searches. Following this, we will present a series of general conjectures, and discuss steps that have been taken towards resolving these.

All work within this chapter is my own, unless otherwise specified.

4.1 Preliminary Definitions

Before examining particular problems, a series of definitions will be required.

4.1.1 k-Staggers

Definition 4.1.1. Let $k \in \mathbb{N}$. Then a *k-stagger* is a PLS P satisfying the following conditions:

- Each row of P contains exactly k entries;
- Each column of P contains exactly k entries;
- Each value $v \in \{1, \ldots, n\}$ appears exactly k times in P.

Example 4.1.2. In the illustration below, P is a 2-stagger of order 6 and Q is a 3-stagger of order 5 (recall that the order of a PLS is the size of its base set).

$$P = \begin{array}{|c|c|c|c|c|c|} \hline 1 & & 3 & & & \\ \hline & & 4 & & & 6 \\ \hline & 4 & & & 6 & \\ \hline & 2 & & 5 & & \\ \hline 5 & & & & & 3 \\ \hline & & & 2 & 1 & \\ \hline \end{array}\,, \quad Q = \begin{array}{|c|c|c|c|c|} \hline & & 3 & 4 & 1 \\ \hline 5 & & 2 & & 3 \\ \hline 4 & 2 & & 1 & \\ \hline 2 & 1 & 5 & & \\ \hline & 4 & & 3 & 5 \\ \hline \end{array}\,.$$

Remark. Recall Definition 1.1.2, in which a PLS P was defined to be a subset of S^3, where S is the base set of P.

Abiding by this terminology, the conditions presented in Definition 4.1 can be rephrased as follows:

- For each $r \in \{1, \ldots, n\}$, exactly k triples in P are of the form (r, x, y);
- For each $c \in \{1, \ldots, n\}$, exactly k triples in P are of the form (y, c, x);
- For each $v \in \{1, \ldots, n\}$, exactly k triples in P are of the form (x, y, v).

Notice that these conditions are symmetrical about rows, columns and values. Thus the principle of symmetry, as described in Section 1.1.1, is applicable not only to PLSs, but also to k-staggers.

Thus, for instance, any theorem regarding the rows of a k-stagger immediately implies a corresonding theorem regarding the columns and another regarding the values found within a k-stagger.

4.1.2 Transversals

Transversals are well defined in the literature.

Definition 4.1.3. Let T be a PLS of order n. Then T is a *transversal* if:

- Each row of T contains exactly one entry;
- Each column of T contains exactly one entry;
- Each value $v \in \{1, \ldots, n\}$ appears exactly once in T.

Remark. Note then that a transversal is simply a 1-stagger.

Example 4.1.4. A transversal of order 5 is shown below.

	3			
			1	
				4
		2		
5				

4.1.3 Orthogonal Latin Squares

Orthogonal latin squares have been well studied. The material in this section is based primarily on [3].

Definition 4.1.5. Let L and M be latin squares of the same order n. Then L and M are said to be *orthogonal* if the following condition is satisfied:

- For all pairs of values $x, y \in \{1, \ldots, n\}$, there is exactly one choice of $r, c \in \{1, \ldots, n\}$ for which cell (r, c) of L contains value x and cell (r, c) of M contains y.

Example 4.1.6. In the illustration below, L and M are orthogonal latin squares of order 4.

$$L = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 2 & 1 & 4 & 3 \\ \hline 3 & 4 & 1 & 2 \\ \hline 4 & 3 & 2 & 1 \\ \hline \end{array}, \quad M = \begin{array}{|c|c|c|c|} \hline 1 & 3 & 4 & 2 \\ \hline 4 & 2 & 1 & 3 \\ \hline 2 & 4 & 3 & 1 \\ \hline 3 & 1 & 2 & 4 \\ \hline \end{array}.$$

For instance, choose $x = 3$ and $y = 4$. Then there is exactly one pair (r, c) for which cell (r, c) of L contains value 3 and cell (r, c) of M contains value 4. This pair is $(r, c) = (1, 3)$.

Remark. Note that the symmetry of Definition 4.1.5 implies that, if L and M are orthogonal, then M and L are also orthogonal.

The following lemma provides a defining property of orthogonal latin squares.

Lemma 4.1.7. *Let L and M be latin squares of the same order n. Then L and M are orthogonal if and only if the following condition is satisfied:*

- *If cells (r_1, c_1) and (r_2, c_2) of L contain the same value, then cells (r_1, c_1) and (r_2, c_2) of M contain different values.*

Note that we do not require the cells of M to contain different values from those in the cells of L; we simply require the two cells of M to contain different values from *each other.*

Example 4.1.8. In Example 4.1.6 above, let $(r_1, c_1) = (3, 1)$ and $(r_2, c_2) = (2, 4)$. Both cells $(3, 1)$ and $(2, 4)$ of L contain the same value, namely 3. So, as expected from Lemma 4.1.7, cells $(3, 1)$ and $(2, 4)$ of M contain different values, namely 2 and 3 respectively.

A proof of Lemma 4.1.7 was not given in [3], and so the following proof is my own.

Proof. Let L and M be latin squares of the same order n.

Say L and M are orthogonal. Furthermore, say cells (r_1, c_1) and (r_2, c_2) of L contain the same value. Let this value be x. Then, if cells (r_1, c_1) and (r_2, c_2) of M contained the same value, say y, we would have two pairs (r, c) for which cell (r, c) of L contains value x and cell (r, c) of M contains value y. However, this contradicts Definition 4.1.5. Thus cells (r_1, c_1) and (r_2, c_2) of M contain different values. Hence L and M satisfy the condition of Lemma 4.1.7.

On the other hand, assume L and M satisfy the condition of Lemma 4.1.7. Furthermore, say there is some choice of values x, y for which there are two pairs (r, c) such that cell (r, c) of L contains value x and cell (r, c) of M contains value y. Let these two pairs be (r_1, c_1) and (r_2, c_2). Then, since both cells (r_1, c_1) and (r_2, c_2) of L contain the same value, the condition of Lemma 4.1.7 implies that cells (r_1, c_1) and (r_2, c_2) of M must contain different values. This is a contradiction, since both contain the same value, namely y.

So, for any choice of values x, y, there is at most one pair (r, c) for which cell (r, c) of L contains value x and cell (r, c) of M contains value y. Since there are exactly n^2 choices of values x, y, there can thus be at most n^2 pairs (r, c) for which cells (r, c) of L and (r, c) of M both contain entries, with equality if and only if the required pair (r, c) exists for all choices of values x, y.

However, since both L and M are latin squares, the number of pairs (r, c) for which cells (r, c) of L and (r, c) of M both contain entries is indeed equal to n^2. Thus, since equality holds, the required pair (r, c) exists for all choices of values x, y.

Hence L and M satisfy the condition of Definition 4.1.5, and so L and M are orthogonal. $\square$

The following theorem then describes for which orders n a pair of orthogonal latin squares can be found.

Theorem 4.1.9. *For every $n \in \mathbb{N} \backslash \{2, 6\}$, a pair of orthogonal latin squares of order n can be found.*

If $n = 2$ or $n = 6$, no pair of orthogonal latin squares of order n exists.

The proof of this theorem is non-trivial, and will not be presented in this thesis. It can be found in [3].

4.2 Existence of k-Staggers

We will now answer the question that asks for which k and n a k-stagger of order n exists. We will show that such a k-stagger exists whenever $k \leq n$.

Theorem 4.2.1. *Let $k, n \in \mathbb{N}$. Then a k-stagger of order n exists if and only if $k \leq n$.*

Proof. First, say $k > n$. Then it is impossible to form a PLS of order n in which each row contains k entries. Thus there is no k-stagger of order n.

Now say $k \leq n$. We will prove the existence of a k-stagger of order n.

Case $n \in \mathbb{N} \backslash \{2, 6\}$:

Say $n \in \mathbb{N} \backslash \{2, 6\}$. Then Theorem 4.1.9 implies the existence of a pair of orthogonal latin squares of order n. Let these be L and M.

For each $x \in \{1, \ldots, n\}$, note that value x must occur in L exactly n times. So let the cells of L containing value x be $(r_{x,1}, c_{x,1})$, …, (r_{x_n}, c_{x_n}). Then Lemma 4.1.7 implies that cells (r_{x_1}, c_{x_1}), …, (r_{x_n}, c_{x_n}) of M must contain different values. Since there are n such cells, these values must be all of $1, \ldots, n$, in some order.

So let P be the PLS created as follows:

- For each $i = 1, \ldots, k$, insert into P the entries in cells $(r_{i,1}, c_{i,1})$, …, (r_{i_n}, c_{i_n}) of M.

Then we claim that P is a k-stagger of order n. This can be seen as follows:

- Since M is a latin square and P contains only entries belonging to M, it follows that P is indeed a PLS (i.e. the row latin and column latin conditions are satisfied).

- It was noted above that, for each i, cells $(r_{i,1}, c_{i,1})$, …, (r_{i_n}, c_{i_n}) of M contain each value in $\{1, \ldots, n\}$ exactly once.

  So, since the entries belonging to k such cell collections have been used to create P, it follows that each value in $\{1, \ldots, n\}$ appears exactly k times in P.

- For each i, cells $(r_{i,1}, c_{i,1})$, …, (r_{i_n}, c_{i_n}) of L are precisely those containing value i. Hence each row contains exactly one of these cells.

  So, since the entries belonging to k such cell collections have been used to create P, it follows that each row of P contains exactly k entries.

- A similar argument shows that each column of P contains exactly k entries.

Hence P is a k-stagger, as required. Since P has order n, it follows that a k-stagger of order n exists.

An example of such a construction is now shown. Let $k = 2$ and $n = 4$. Let L and M be as shown below. Note that L and M are orthogonal.

$$L = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 2 & 1 & 4 & 3 \\ \hline 3 & 4 & 1 & 2 \\ \hline 4 & 3 & 2 & 1 \\ \hline \end{array}, \quad M = \begin{array}{|c|c|c|c|}\hline 1 & 3 & 4 & 2 \\ \hline 4 & 2 & 1 & 3 \\ \hline 2 & 4 & 3 & 1 \\ \hline 3 & 1 & 2 & 4 \\ \hline \end{array}.$$

Then, since $k = 2$, we list the locations of entries in L containing values 1 and 2. These locations are shown below.

$$\begin{aligned}
(r_{1,1}, c_{1,1}) &= (1,1), & (r_{2,1}, c_{2,1}) &= (1,2),\\
(r_{1,2}, c_{1,2}) &= (2,2), & (r_{2,2}, c_{2,2}) &= (2,1),\\
(r_{1,3}, c_{1,3}) &= (3,3), & (r_{2,3}, c_{2,3}) &= (3,4),\\
(r_{1,4}, c_{1,4}) &= (4,4), & (r_{2,4}, c_{2,4}) &= (4,3).
\end{aligned}$$

The corresponding entries of M are used to form P, as follows:

$$P = \begin{array}{|c|c|c|c|}\hline 1 & 3 & & \\ \hline 4 & 2 & & \\ \hline & & 3 & 1 \\ \hline & & 2 & 4 \\ \hline \end{array}\,.$$

Note then that P is indeed a 2-stagger of order 4.

Case $n = 2$:

PLSs P_1, P_2, as illustrated below, form a 1-stagger of order 2 and a 2-stagger of order 2 respectively.

$$P_1 = \begin{array}{|c|c|}\hline 1 & \\ \hline & 2 \\ \hline \end{array}\,, \quad P_2 = \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array}\,.$$

Thus, for $k \leq 2$, a k-stagger of order $n = 2$ exists.

Case $n = 6$:

Again, for each $k \in \{1, \ldots, 6\}$, the PLS P_k shown below is a k-stagger of order 6. These were discovered with the aid of a computer search.

$$P_1 = \begin{array}{|c|c|c|c|c|c|}\hline 1 & & & & & \\ \hline & 2 & & & & \\ \hline & & 3 & & & \\ \hline & & & 4 & & \\ \hline & & & & 5 & \\ \hline & & & & & 6 \\ \hline \end{array}\,, \quad P_2 = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & & & & \\ \hline & 2 & 3 & & & \\ \hline & & 4 & 5 & & \\ \hline & & & 6 & 1 & \\ \hline & & & & 2 & 3 \\ \hline 5 & & & & & 4 \\ \hline \end{array}\,,$$

$$P_3 = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & & & & 5 \\ \hline 1 & 2 & 3 & & & \\ \hline & & 4 & 5 & 3 & \\ \hline & 4 & & 6 & 1 & \\ \hline & & 6 & & 2 & 3 \\ \hline 5 & & & 2 & & 4 \\ \hline \end{array}\,, \quad P_4 = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & & & 4 & 5 \\ \hline 1 & 2 & 3 & & & 6 \\ \hline 2 & & 4 & 5 & 3 & \\ \hline & 4 & 5 & 6 & 1 & \\ \hline & & 6 & 1 & 2 & 3 \\ \hline 5 & 3 & & 2 & & 4 \\ \hline \end{array}\,,$$

$$P_5 = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & & 3 & 4 & 5 \\ \hline 1 & 2 & 3 & & 5 & 6 \\ \hline 2 & 6 & 4 & 5 & 3 & \\ \hline & 4 & 5 & 6 & 1 & 2 \\ \hline 4 & & 6 & 1 & 2 & 3 \\ \hline 5 & 3 & 1 & 2 & & 4 \\ \hline \end{array}\,, \quad P_6 = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 2 & 6 & 4 & 5 & 3 & 1 \\ \hline 3 & 4 & 5 & 6 & 1 & 2 \\ \hline 4 & 5 & 6 & 1 & 2 & 3 \\ \hline 5 & 3 & 1 & 2 & 6 & 4 \\ \hline \end{array}\,.$$

Thus, for $k \leq 6$, a k-stagger of order $n = 6$ exists. □

4.3 Completions

The problem with which the remainder of this chapter will be occupied can now be stated as follows.

Problem 4.3.1. *For which values $n, k \in \mathbb{N}$, $k \leq n$, are* all *k-staggers of order n valid?*

4.3.1 1-Staggers

The specific case of Problem 4.3.1 corresponding to $k = 1$ will now be discussed.

Definition 4.3.2. Let $n \in \mathbb{N}$. Then the *primary transversal* of order n is defined to be the PLS T constructed as follows:

- For each $i \in \{1, \ldots, n\}$, place value i in cell (i, i) of T.

Note that, since each entry contains a different value, the row latin and column latin conditions are satisfied. Thus T is a PLS. Furthermore, each row, column and value is used exactly once. Thus T is indeed a transversal, as its name suggests.

Example 4.3.3. The primary transversal of order 4 is shown below.

1			
	2		
		3	
			4

Lemma 4.3.4. *Let P be a 1-stagger of order n. Then P is valid if and only if the primary transversal of order n is valid.*

Proof. Note that, by definition of a 1-stagger, each row, column and value is used in exactly one entry of P. In particular, this means that values $1, \ldots, n$ appear in different rows of P.

So permute the rows of P using some permutation α, such that value i appears in row i, for all i. Similarly, the columns of P can then be permuted using some permutation β, such that value i appears in column i, for all i.

Let the resulting PLS be T. Then each row, column and value is used in exactly one entry of T, since this is also true of P. Hence T contains exactly n entries. Furthermore, for each i, value i appears in cell (i, i) of T. Thus the n entries in T are the same n entries that appear in the primary transversal of order n. So T is in fact the primary transversal of order n.

Now say P has some completion L. Then applying permutations α and β to the rows and columns respectively of L produces a completion of T.

On the other hand, let T have some completion L'. Then applying permutations α^{-1} and β^{-1} to the rows and columns respectively of L' produces a completion of P.

Hence P is valid if and only if T (the primary transversal of order n) is valid. □

The following corollary then follows from Lemma 4.3.4.

Corollary 4.3.5. *Let P be a 1-stagger of order n. Then all 1-staggers of order n are valid if and only if P is valid.*

Proof. Say all 1-staggers of order n are valid. Then, since P is a 1-stagger of order n, it follows that P is valid.

On the other hand, say P is valid. Let T be the primary transversal of order n. Furthermore, let Q be any 1-stagger of order n. From Lemma 4.3.4, since P is valid, we know that T is valid. Using Lemma 4.3.4 again, since T is valid, we then know that Q is valid. So all 1-staggers of order n are valid, as required. $\square$

We can now prove the following validity result for 1-staggers.

Theorem 4.3.6. *If $n \in \mathbb{N}\backslash\{2\}$, then every 1-stagger of order n is valid.*
However, all 1-staggers of order 2 are invalid.

Proof. We will split into cases, based upon the possible values of n.

Case $n \in \mathbb{N}\backslash\{2, 6\}$:

From Theorem 4.1.9, there are orthogonal latin squares L and M of order n. Note that the value 1 appears in L exactly n times. Let the cells of L in which value 1 appears be (r_1, c_1), …, (r_n, c_n).

Then, by Lemma 4.1.7, the values in cells (r_1, c_1), …, (r_n, c_n) of M are all distinct. Thus these values are $1, \ldots, n$, in some order.

Furthermore, the row latin and column latin conditions of L imply that rows $r_1, \ldots, r_n$ are distinct and that columns $c_1, \ldots, c_n$ are distinct. Thus these collections of rows and columns are also $1, \ldots, n$, in some order.

So define the PLS T of order n as follows:

- For each $i \in \{1, \ldots, n\}$, fill cell (r_i, c_i) of T with the value appearing in cell (r_i, c_i) of M.

Then each row, column and value appears exactly once in the entries of T. So T is a 1-stagger of order n.

Furthermore, since each entry of T lies also in M, we see that the latin square M is a completion of T. Hence T is valid.

So, from Corollary 4.3.5, all 1-staggers of order n are valid.

An illustration of this procedure will now be given, for the case $n = 4$. Orthogonal latin squares L and M of order 4 are shown below.

$$L = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 2 & 1 & 4 & 3 \\ \hline 3 & 4 & 1 & 2 \\ \hline 4 & 3 & 2 & 1 \\ \hline \end{array}, \quad M = \begin{array}{|c|c|c|c|}\hline 1 & 3 & 4 & 2 \\ \hline 4 & 2 & 1 & 3 \\ \hline 2 & 4 & 3 & 1 \\ \hline 3 & 1 & 2 & 4 \\ \hline \end{array}.$$

The cells in which value 1 appears in L are:

$$\begin{aligned} (r_1, c_1) &= (1, 1) \\ (r_2, c_2) &= (2, 2) \\ (r_3, c_3) &= (3, 3) \\ (r_4, c_4) &= (4, 4). \end{aligned}$$

Thus T is defined by the entries of M appearing in these cells, as shown below.

$$T = \begin{array}{|c|c|c|c|}\hline 1 & & & \\ \hline & 2 & & \\ \hline & & 3 & \\ \hline & & & 4 \\ \hline \end{array}.$$

It can then be seen that T is a 1-stagger of order 4, and that M is a completion of T.

Case $n = 6$:

The following example was obtained using a computer search. In the illustration below, T is a 1-stagger of order 6, and M is a completion of T. So T is valid. Thus, from Corollary 4.3.5, all 1-staggers of order 6 are valid.

$$T = \begin{array}{|c|c|c|c|c|c|}\hline & & 2 & & & \\ \hline & & & 4 & & \\ \hline & & & & & 1 \\ \hline 3 & & & & & \\ \hline & 5 & & & & \\ \hline & & & & 6 & \\ \hline \end{array}, \quad M = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 2 & 6 & 4 & 5 & 3 & 1 \\ \hline 3 & 4 & 5 & 6 & 1 & 2 \\ \hline 4 & 5 & 6 & 1 & 2 & 3 \\ \hline 5 & 3 & 1 & 2 & 6 & 4 \\ \hline \end{array}.$$

Case $n = 2$:

Consider T, shown below, which is a 1-stagger of order 2.

$$T = \begin{array}{|c|c|}\hline 1 & \\ \hline & 2 \\ \hline \end{array}.$$

T has no completion, since cell $(1, 2)$ cannot contain either value 1 or 2, by the row latin and column latin conditions respectively.

Thus T is invalid, and so Corollary 4.3.5 implies that all 1-staggers of order 2 are invalid. □

4.3.2 2-Staggers

We will now examine the specific case of Problem 4.3.1 corresponding to $k = 2$. Unlike the case $k = 1$, a complete solution has not been obtained. However, a number of partial results will be presented.

Lemma 4.3.7. *All 2-staggers of order 2 are valid.*

Proof. This is trivial, since any 2-stagger of order 2 must be a complete latin square (since each row contains 2 entries), and thus forms its own completion. □

In fact, the same argument produces the following (just as trivial!) result:

Lemma 4.3.8. *Let $n \in \mathbb{N}$. Then all n-staggers of order n are valid.*

A computer program was written, designed to find all invalid k-staggers of order n, for given values of k and n. The source code is split across several files, all of which are provided in Appendix A.

In particular, searches were performed to find all invalid 2-staggers of order n, using a number of different orders n. The results of these searches are now presented.

Lemma 4.3.9. *An invalid 2-stagger of order n exists, for all $n \in \{3, \ldots, 7\}$.*

Proof. For each $n \in \{3, \dots, 7\}$, the 2-stagger P_n is shown below. In each case, P_n has order n.

$$P_3 = \begin{array}{|c|c|c|}\hline 1 & 2 & \\ \hline & 1 & 3 \\ \hline 3 & & 2 \\ \hline \end{array}\,, \quad P_4 = \begin{array}{|c|c|c|c|}\hline 4 & 1 & & \\ \hline 1 & & 4 & \\ \hline & 2 & & 3 \\ \hline & & 3 & 2 \\ \hline \end{array}\,,$$

$$P_5 = \begin{array}{|c|c|c|c|c|}\hline 5 & 1 & & & \\ \hline 1 & & 5 & & \\ \hline & 2 & & 3 & \\ \hline & & 2 & & 4 \\ \hline & & & 4 & 3 \\ \hline \end{array}\,, \quad P_6 = \begin{array}{|c|c|c|c|c|c|}\hline 6 & 1 & & & & \\ \hline 1 & & 6 & & & \\ \hline & 2 & & 3 & & \\ \hline & & 2 & & 3 & \\ \hline & & & 4 & & 5 \\ \hline & & & & 5 & 4 \\ \hline \end{array}\,,$$

$$P_7 = \begin{array}{|c|c|c|c|c|c|c|}\hline 7 & 1 & & & & & \\ \hline 1 & & 7 & & & & \\ \hline & 2 & & 3 & & & \\ \hline & & 3 & 2 & & & \\ \hline & & & & 4 & 5 & \\ \hline & & & & 5 & & 6 \\ \hline & & & & & 6 & 4 \\ \hline \end{array}\,.$$

Furthermore, in each of the above cases, a thorough computer search failed to find any completion of P_n. Thus each 2-stagger shown above is invalid.

Hence an invalid 2-stagger exists for each of the required orders. ☐

Theorem 4.3.10. *All 2-staggers of order 8 are valid.*

Proof. A thorough computer search failed to find any 2-stagger of order 8 for which a completion did not exist. ☐

Theorem 4.3.10 leads to the following conjecture:

Conjecture 4.3.11 (First Conjecture)**.** *If $n \geq 8$, then every 2-stagger of order n is valid.*

A computer search was begun on the case $n = 9$, but the complexity of the problem (in terms of required execution time) has thus far made the search infeasible.

4.3.3 Potential Proof of First Conjecture

All attempts to prove Conjecture 4.3.11 have to date proven unfruitful. However, the method that has appeared most promising will now be outlined. Before presenting this method, however, a preliminary result will be required.

Lemma 4.3.12. *Let P be a k-stagger of order n, where $k < n$. Let T be a transversal of order n, such that no entry of T shares the same cell as any entry of P.*

Let $P' = P \cup T$. If P' satisfies the row latin and column latin conditions, then P' is a $(k+1)$-stagger of order n.

Example 4.3.13. In the illustration below, P is a 2-stagger of order 5 and T is a transversal of order 5. Furthermore, no entry of T shares the same cell as any entry of P.

$$P = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & & & \\ \hline & 3 & 4 & & \\ \hline & & 5 & 1 & \\ \hline & & & 2 & 3 \\ \hline 5 & & & & 4 \\ \hline \end{array}\,, \quad T = \begin{array}{|c|c|c|c|c|}\hline & & & 4 & \\ \hline & & & & 5 \\ \hline 1 & & & & \\ \hline & 2 & & & \\ \hline & & 3 & & \\ \hline \end{array}\,.$$

The union P' is shown below. It can be seen that P' satisfies the row latin and column latin conditions. Furthermore, it can also be seen that P' is a 3-stagger of order 5, as expected from Lemma 4.3.12.

$$P' = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & & 4 & \\ \hline & 3 & 4 & & 5 \\ \hline 1 & & 5 & 1 & \\ \hline & 2 & & 2 & 3 \\ \hline 5 & & 3 & & 4 \\ \hline \end{array}\,.$$

A proof of Lemma 4.3.12 is now given.

Proof. Since no entry of T shares the same cell as any entry of P, we do not have two values attempting to occupy the same cell of P'. Furthermore, since P' satisfies both the row latin and column latin condition, it follows that P' is indeed a PLS of order n.

Now, since each row, column and value is used in precisely k entries of P and precisely 1 entry of T, it follows that each row, column and value is used in exactly $(k+1)$ entries of P'. Thus P' is a $(k+1)$-stagger of order n. □

The proposed method of proof for Conjecture 4.3.11 is then as follows.

Proposed method of proof:

Let P be a 2-stagger of order n, where $n \geq 8$. Then a series of PLSs $P_2, \ldots, P_n$ is constructed as follows:

- Let $P_2 = P$.
- For each $k \in \{3, \ldots, n\}$, let P_k be constructed from P_{k-1} by adding a transversal, as described in Lemma 4.3.12.

Then, for each $k \in \{2, \ldots, n\}$, the following properties hold:

- P_k is an extension of P;
- P_k is an k-stagger of order n.

In particular, P_n is an n-stagger of order n, and so is a latin square (since each row contains n elements). So P_n is a completion of P, and thus P is valid.

The primary difficulty in the above method of proof is in utilising Lemma 4.3.12, which requires the row latin and column latin conditions to hold for each newly created PLS. In general, it is difficult to find suitable transversals T, for use with Lemma 4.3.12, that ensure that these row latin and column latin conditions hold.

Thus in general, some extra property would be required of the intermediate PLSs $P_3, \ldots, P_{k-1}$, in order to provide the extra conditions necessary to use Lemma 4.3.12

(such a property would be used in a fashion similar to the property of "cunningness, as described in Construction 3.3.9).

However, a result will now be presented that describes a class of situations in which Lemma 4.3.12 can be used.

Lemma 4.3.14. *Let P be a k-stagger of order n. Let (r_1, c_1), ..., (r_n, c_n) be any collection of empty cells of P, such that each row and column $1, \ldots, n$ is used precisely once in this collection.*

If $n \geq 4k$, then cells (r_1, c_1), ..., (r_n, c_n) of P can be filled to produce an extension P' of P, in such a manner that P' is a $(k+1)$-stagger.

Proof. For each $i \in \{1, \ldots, n\}$, define the following sets.

- Let R_i be the set of values appearing in row r_i of P;
- Let C_i be the set of values appearing in column c_i of P;
- Let $S_i = S \backslash (R_i \cup C_i)$, where $S = \{1, \ldots, n\}$.

Thus, for each i, R_i and C_i represent the sets of values that may not be placed in cell (r_i, c_i) without violating the row latin or column latin condition respectively. So S_i represents the set of values that may be placed in cell (r_i, c_i) without violating the row latin or column latin condition.

For illustration, consider the 2-stagger P of order 8 given below. Note that $8 \geq 4 \cdot 2$.

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

Locations (r_1, c_1), ..., (r_n, c_n) are marked with asterisks $(*)$, and are as follows.

$$\begin{array}{ll}
(r_1, c_1) = (1, 6), & (r_5, c_5) = (5, 4), \\
(r_2, c_2) = (2, 8), & (r_6, c_6) = (6, 1), \\
(r_3, c_3) = (3, 5), & (r_7, c_7) = (7, 7), \\
(r_4, c_4) = (4, 3), & (r_8, c_8) = (8, 2).
\end{array}$$

The corresponding sets are then:

$$\begin{array}{lll}
R_1 = \{1, 5\}, & C_1 = \{1, 4\}, & S_1 = \{2, 3, 6, 7, 8\}, \\
R_1 = \{3, 6\}, & C_1 = \{4, 5\}, & S_1 = \{1, 2, 7, 8\}, \\
R_1 = \{1, 2\}, & C_1 = \{6, 8\}, & S_1 = \{3, 4, 5, 7\}, \\
R_1 = \{4, 6\}, & C_1 = \{5, 8\}, & S_1 = \{1, 2, 3, 7\}, \\
R_1 = \{3, 4\}, & C_1 = \{6, 7\}, & S_1 = \{1, 2, 5, 8\}, \\
R_1 = \{2, 7\}, & C_1 = \{1, 3\}, & S_1 = \{4, 5, 6, 8\}, \\
R_1 = \{7, 8\}, & C_1 = \{2, 3\}, & S_1 = \{1, 4, 5, 6\}, \\
R_1 = \{5, 8\}, & C_1 = \{2, 7\}, & S_1 = \{1, 3, 4, 6\}.
\end{array}$$

Say a SDR $\langle s_1, \ldots, s_n \rangle$ exists for sets $S_1, \ldots, S_n$. Then we form P' by placing value s_i in cell (r_i, c_i), for each $i \in \{1, \ldots, n\}$.

Since each row and column is used precisely once in the collection (r_1, c_1), …, (r_n, c_n), it follows that each row and column has precisely one new entry placed within it. Furthermore, since representatives $s_1, \ldots, s_n$ are distinct, it follows that each value $1, \ldots, n$ appears in precisely one of the n new entries.

Thus the new entries form a transversal T of order n, and we have $P' = P \cup T$. Furthermore, since $s_i \in S_i$ for each i and since the values $s_1, \ldots, s_n$ are distinct, we see that the row latin and column latin conditions are satisfied for P'.

Since $k \in \mathbb{N}$, we have $n \geq 4k > k$. Finally, since cells (r_1, c_1), …, (r_n, c_n) of P are empty, we see that no entry of T shares the same cell as any entry of P.

Thus the conditions for Lemma 4.3.12 are satisfied, and so P' is a $(k+1)$-stagger of order n. Hence cells (r_1, c_1), …, (r_n, c_n) of P have been filled to produce the extension P' of P, in which P' is a $(k+1)$-stagger, as required.

Continuing with our example above, a SDR for sets $S_1, \ldots, S_8$ is $\langle 3, 7, 4, 1, 2, 8, 5, 6 \rangle$. These values can then be placed into cells (r_1, c_1), …, (r_8, c_8) respectively, producing P' as shown below. The transversal T is also given.

$$P' = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & & 5 & & & 3 & & \\\hline & & & 6 & & & 3 & 7 \\\hline & 2 & & & 4 & 1 & & \\\hline & & 1 & & 6 & 4 & & \\\hline 3 & & & 2 & & & & 4 \\\hline 8 & 7 & & & & & 2 & \\\hline & & 8 & 7 & & & 5 & \\\hline & 6 & & & 8 & & & 5 \\\hline \end{array}\,, \quad T = \begin{array}{|c|c|c|c|c|c|c|c|}\hline & & & & & 3 & & \\\hline & & & & & & & 7 \\\hline & & & & 4 & & & \\\hline & & 1 & & & & & \\\hline & & & 2 & & & & \\\hline 8 & & & & & & & \\\hline & & & & & & 5 & \\\hline & 6 & & & & & & \\\hline \end{array}\,.$$

Note that the PLS P' above is indeed a 3-stagger, as expected.

Existence of a SDR:

All that remains then is to prove the existence of a SDR for sets $S_1, \ldots, S_n$. For this, we will (surprise!) use Hall's Theorem.

Choose any $m \in \{0, \ldots, n\}$ and any m sets from the above collection. Let these be $S_{i_1}, \ldots, S_{i_m}$. If $m = 0$, the union of 0 sets contains at least 0 elements, as required.

Otherwise, we will take two cases.

- *If $m \leq n - 2k$:*

  Notice that, for each i, $|R_i| = |C_i| = k$, since each row and column of a k-stagger contains precisely k elements. Thus

$$\begin{aligned} |S_i| &= |S \backslash (R_i \cup C_i)| \\ &\geq |S| - |R_i| - |C_i| \\ &= n - 2k \\ &\geq m. \end{aligned}$$

  So, since $|S_i| \geq m$ for each i, we have $|S_{i_1} \cup \ldots \cup S_{i_m}| \geq m$.

- *If $m > n - 2k$:*

Since each value x appears in precisely k rows and k columns of P, we see that x belongs to precisely k of the sets R_j and k of the sets C_j. Hence x belongs to at most $2k$ of the sets $(R_j \cup C_j)$, and so belongs to at least $n-2k$ of the sets $S_j = S \backslash (R_j \cup C_j)$.

In particular, this implies that each value x belongs to at least

$$(n-2k)-(n-m) = m-2k > (n-2k)-2k = n-4k \tag{i}$$

of the sets $S_{i_1}, \ldots, S_{i_m}$. However, since $n \geq 4k$, this in turn implies that each value x belongs to at least one of the sets $S_{i_1}, \ldots, S_{i_m}$ (notice the strict inequality in equation i).

So each value x belongs to the union $S_{i_1} \cup \ldots \cup S_{i_m}$. Thus

$$|S_{i_1} \cup \ldots \cup S_{i_m}| = n \geq m.$$

In either case, we have $|S_{i_1} \cup \ldots \cup S_{i_m}| \geq m$. So, by Hall's Theorem, a SDR exists for $S_1, \ldots, S_n$. □

4.3.4 A General Conjecture

Recall the proposed construction presented in Section 4.3.3. We noted that some extra property may be required of our intermediate PLSs $P_3, \ldots, P_{n-1}$.

Since Lemma 4.3.14 requires no such extra properties, aside from the condition $n \geq 4k$, it follows that this lemma may be useful in providing the first step of our construction. That is, the construction proceeds as follows:

- Begin with $P = P_2$;
- Use Lemma 4.3.14 to extend P_2 to P_3, where P_3 has some extra property;
- For each $k \in \{4, \ldots, n\}$, use Lemma 4.3.12, along with the extra property of P_{k-1}, to extend P_{k-1} to P_k, which also has this extra property.

However, such a method of construction is possible only if Lemma 4.3.14 is applicable for $k = 2$. This requires $n \geq 4k = 8$.

But recall that $n \geq 8$ was the bound proposed in Conjecture 4.3.11! This way in which our proposed bound $n \geq 8$ "falls out" of the above discussion provides support for our proposed method of proof described in Section 4.3.3.

In fact, we may now use this bound $n \geq 4k$ to formulate a more general conjecture:

Conjecture 4.3.15. *If $n \geq 4k$, then every k-stagger of order n is valid.*

The method of proving Conjecture 4.3.15 would then follow the similar lines to the method described in Section 4.3.3 and discussed above. Lemma 4.3.14 would be used to extend any given k-stagger to a $(k+1)$-stagger that satisfies some extra property, and this in turn would be extended inductively via Lemma 4.3.12 until a latin square was produced.

To conclude this chapter, we shall examine the particular cases of Conjecture 4.3.15 corresponding to $k = 2$ and $k = 1$.

If $k = 2$, Conjecture 4.3.15 reduces to Conjecture 4.3.11, as discussed above.

If $k = 1$, Conjecture 4.3.15 requires all 1-staggers of orders $n \geq 4$ to be valid. In fact, Theorem 4.3.6 shows that all 1-staggers of orders $n \geq 3$ are valid. Thus, whilst Conjecture 4.3.15 is correct in the case $k = 1$, it does not reflect the lowest possible bound for n.

Chapter 5

Conclusion

In Chapter 1, a series of preliminary definitions and results were presented. In addition, Philip Hall's Theorem, which describes the conditions under which a SDR exists for a given collection of sets, was stated and proved.

Chapter 2 saw the introduction of our first completion theorem. This was Theorem 2.1.3, due to Marshall Hall, which states that any $r \times n$ latin rectangle is valid. Two proofs were presented. The first, based upon the literature, utilises Hall's Theorem (regarding SDRs). However, since Hall's Theorem is an existence theorem, this proof does not supply a direct construction for completing an arbitrary latin rectangle. It is conceivable that such a direct construction may be required for computational purposes. So a second proof was provided. This was my own proof, and contains within it a direct construction for completing an arbitrary latin rectangle.

Following this, Chapter 3 is devoted to the proof of Evans' Conjecture, which states that any PLS of order n containing at most $n-1$ entries is valid. In places, the proof follows those presented in [2] and [5]. However, other sections of the proof (most notably the proof that Construction 3.3.9 is correct) are my own.

Finally, Chapter 4 deals with the problem of completing k-staggers, and asks the following question:

For which values of k and n are all k-staggers of order n valid?

Theorem 4.2.1 shows, through the use of orthogonal latin squares, that k-staggers of order n exist for all k, n satisfying $k \leq n$.

The completion problem is then discussed for the case $k = 1$, resulting in Theorem 4.3.6, which shows that a 1-stagger of order n is valid if and only if $n \neq 2$. Following this, the case $k = 2$ is examined. No complete solution is obtained for this case. However, it is shown through the use of computer searches that invalid 2-staggers exist for all orders $n \in \{3, \dots, 7\}$, and that all 2-staggers of order 8 are valid. We thus propose Conjecture 4.3.11, which states that all 2-staggers of order $n \geq 8$ are valid.

Section 4.3.3 describes what has appeared to be the most promising method of attack in attempting to prove this conjecture. Any given 2-stagger of order $n \geq 8$ is extended to a 3-stagger through the use of Lemma 4.3.12. This in turn is extended to a 4-stagger, and this process continues in an inductive fashion until an n-stagger is produced, which is a latin square.

Lemma 4.3.14 is then proven, which provides the means for performing the first step of this process, namely that of extending the initial 2-stagger to a 3-stagger. From the condition $n \geq 4k$ presented in this lemma, a more general conjecture is proposed. This is Conjecture 4.3.15, which states that all k-staggers of order $n \geq 4k$ are valid.

This conjecture is then verified for $k = 1$, although in this case, $n \geq 4k = 4$ does not represent the lowest possible bound for n. For $k = 2$, Conjecture 4.3.15 agrees with the computational results, and in fact reduces to Conjecture 4.3.11. Furthermore, if Conjecture 4.3.11 is in fact correct, Lemma 4.3.9 shows that $n \geq 8$ is indeed the lowest bound possible.

It is not clear whether Conjecture 4.3.15 is true for $k \geq 3$, and if so, whether $n \geq 4k$ represents the best possible bound. It may be that the case $k = 1$, for which $n \geq 4k$ is not the lowest bound possible, simply represents an exceptional case. Small cases for which exceptional properties hold are not uncommon in combinatorial problems.

For $k \geq 3$, the complexity of searches, in terms of execution time, makes a computer search infeasible. The currently existing search program, to some extent, takes into account "isomorphisms" of PLSs produced by permuting rows, columns and value labels. That is, in some cases for which P and Q are PLSs obtained from one another in such a fashion, the computer will only examine one of P and Q.

In order to improve the efficiency of searching, future work in this field may including modifying the search algorithm in order to take into account a larger range of such "isomorphisms". Other directions for future work may include further investigation into the method of proof outlined in Section 4.3.3, and into properties of transversals, which are used by Lemma 4.3.12 in extending k-staggers to $(k + 1)$-staggers.

Bibliography

[1] Lindner, C.C., *A survey of finite embedding theorems for partial latin squares and quasigroups*, In Graphs and Combinatorics, Lecture Notes in Mathematics, **406** (1974), 109–152, Springer-Verlag, Berlin.

[2] Lindner, C.C., *Embedding theorems for partial latin squares*, In Latin Squares: New Developments in the Theory and Applications, editors Denes, J. & Keedwell, A.D. (1991), 217–265, North-Holland, Amsterdam.

[3] Lindner, C.C., *MP475 Lecture Notes*, The University of Queensland, 1995.

[4] Lindner, C.C., *On completing latin rectangles*, Canadian Mathematical Bulletin, **13** (1970), 65–68.

[5] Smetaniuk, B., *A new construction on latin squares - I: A proof of the Evans Conjecture*, Ars Combinatoria, **11** (1981), 155–172.

[6] Williams, S., *MP384 Lecture Notes*, The University of Queensland, 1996.

Appendix A

Computer Search Source Code

Recall from Section 4.3.2 that a computer program was written to find all invalid k-staggers of order n, for given values of k and n. The source code for this program is provided in this appendix.

The program was written using the language C++, and is spread across several source files. Each source file begins with a comment that describes its purpose. When run, the program executes the *main()* function, which is contained in the source file `stagger.cc`.

A.1 Source for *boolean.h*

```
// ---------
// Boolean.h
// ---------
//
// Defines a boolean data type.

#ifndef __BOOLEAN_H
#define __BOOLEAN_H

typedef int boolean;
#define True 1
#define False 0

#endif
```

A.2 Source for *tset.h*

```
// ------
// TSet.h
// ------
//
// Provides functions for dealing with partial latin squares.

#ifndef __TSET_H
#define __TSET_H

#include "boolean.h"

// Triple class: A (value, row, column) triple.
```

```
class Triple
        {
        public:
                int v[3];

                Triple() {}
                Triple(int a, int b, int c) { v[0] = a; v[1] = b; v[2] = c; }

                int operator [] (int pos) { return v[pos]; }
        };

// If t is a triple, then t[0], t[1] and t[2] represent the value, row and
// column of t respectively.

#define SQR(a,b,c) sqr[a][(b)+(n*(c))]

// TSet class: A partial latin square.

class TSet
        {
        protected:
                int n;          // Order
                int t;          // Number of triples stored
                int *sqr[3];    // Representation of square

                boolean validfrom(int i,int j);
                int solnsfrom(int i,int j);
                void copytset(TSet &t2);

        public:
                // --- Constructors and Destructor
                TSet(int order);
                   // Creates empty TSet.
                TSet(TSet &t2);
                   // Create TSet equal to t2.
                virtual ~TSet();
                // --- Operations
                boolean add(Triple&);
                   // Add an entry.
                boolean remove(Triple&);
                   // Remove an entry.
                void empty();
                   // Remove all entries.
                void copy(TSet &t2);
                   // Make a copy of t2.
                // --- Tests
                boolean valid();
                   // Is this PLS valid?
                boolean full()
                { return (t == n*n); }
                   // Is this an entire latin square?
                // --- Calculations
                int solns();
                   // Find number of completions.
                // --- Info Requests
                int order()
```

```
                { return n; }
                   // Find order of PLS.
                int triples()
                { return t; }
                   // Find number of entries.
                int lookup(int i,int j,int sq=0)
                   { return SQR(sq,i,j); }
                   // lookup(i,j) will find the value
                   //    in row i,col j.
                   // Triples are stored as (t[0],t[1],t[2]).
                   // sq represents which place (0,1,2) of the triple
                   //    is to be looked up.
                   // i represents place (1+sq), j represents place
                   //    (2+sq), where place numbers are taken mod 3.
        };

#endif
```

A.3 Source for *tset.cc*

```
// Implementation of functions described in TSet.h.

#include "tset.h"

TSet::TSet(int order)
{
n = order;

int i;
for (i=0;i<3;i++)
   sqr[i] = new int[n*n];

empty();
}

TSet::TSet(TSet &t2)
{
copytset(t2);
}

void TSet::copy(TSet &t2)
{
int i;
for (i=0;i<3;i++)
   delete[] sqr[i];
copytset(t2);
}

void TSet::copytset(TSet &t2)
{
n = t2.n;
t = t2.t;

int i;
for (i=0;i<3;i++)
```

```
   sqr[i] = new int[n*n];

int j,l;
l = n*n;
for (i=0;i<3;i++)
   for (j=0;j<l;j++)
      sqr[i][j] = t2.sqr[i][j];
}

TSet::~TSet()
{
int i;
for (i=0;i<3;i++)
   delete[] sqr[i];
}

boolean TSet::add(Triple &tr)
{
if (!(SQR(0,tr[1],tr[2])==-1 && SQR(1,tr[2],tr[0])==-1 &&
        SQR(2,tr[0],tr[1])==-1))
   return False;

// Triple satisfies row latin and column latin conditions.
SQR(0,tr[1],tr[2]) = tr[0];
SQR(1,tr[2],tr[0]) = tr[1];
SQR(2,tr[0],tr[1]) = tr[2];
t++;
return True;
}

boolean TSet::remove(Triple &tr)
{
if (SQR(0,tr[1],tr[2]) != tr[0])
   return False;

// Triple is actually present.
SQR(0,tr[1],tr[2]) = -1;
SQR(1,tr[2],tr[0]) = -1;
SQR(2,tr[0],tr[1]) = -1;
t--;
return True;
}

void TSet::empty()
{
t=0;
int i,j,l;
l = n*n;
for (i=0;i<3;i++)
   for (j=0;j<l;j++)
      sqr[i][j] = -1;
}

boolean TSet::valid()
{
if (t < n) return True;
```

```
return validfrom(0,0);
}

boolean TSet::validfrom(int i,int j)
{
// Assumes all items before (i,j) have been placed.
int k;
while (i < n)
   {
   while (j < n)
      {
      if (SQR(0,i,j) == -1)
         {
         for (k=0;k<n;k++)
            {
            Triple tr(k,i,j);
            if (add(tr))
               {
               if (validfrom(i,j+1))
                  {
                  remove(tr);
                  return True;
                  }
               remove(tr);
               }
            }
         return False;
         }
      j++;
      }
   j=0;
   i++;
   }
return True;
}

int TSet::solns()
{
return solnsfrom(0,0);
}

int TSet::solnsfrom(int i,int j)
{
// Assumes all items before (i,j) have been placed.
int k;
while (i < n)
   {
   while (j < n)
      {
      if (SQR(0,i,j) == -1)
         {
         int ans=0;
         for (k=0;k<n;k++)
            {
            Triple tr(k,i,j);
            if (add(tr))
```

```
                {
                ans += solnsfrom(i,j+1);
                remove(tr);
                }
             }
          return ans;
          }
       j++;
       }
   j=0;
   i++;
   }
return 1;
}
```

A.4 Source for *output.h*

```
// --------
// OutSet.h
// --------
//
// Outputs a partial latin square to the screen or output file.

#ifndef __TSET_OUTPUT_H
#define __TSET_OUTPUT_H

#include "tset.h"

#include <iostream.h>

void outset(TSet &t, ostream &o);
   // Outputs the PLS t to the output stream o.

#endif
```

A.5 Source for *output.cc*

```
// Implements functions described in Output.h.

#include "output.h"

void outset(TSet &t, ostream &o)
{
int i,j,k;
int n = t.order();
for (i=0;i<n;i++)
   {
   for (j=0;j<n;j++)
      {
      k = t.lookup(i,j);
      if (k==-1)
         o << '.';
      else
         o << k;
      }
```

```
      o << '\n';
   }
}
```

A.6 Source for *stagfunc.h*

```
// ----------
// StagFunc.h
// ----------
//
// Defines a type of function that may be performed upon staggers.

#ifndef __STAGFUNC_H
#define __STAGFUNC_H

typedef void (*StaggerFunc)(TSet&, void *args);

#endif
```

A.7 Source for *allstag.h*

```
// ---------
// AllStag.h
// ---------
//
// Searches through all staggers of a given size
// and performs a given function upon these.
//
// If two staggers are equivalent by row, column or value
// permutations, then this algorithm may only find of them.
//
// However, at least one stagger from each such equivalence
// class is guaranteed to be found.

#ifndef __ALLSTAG_H
#define __ALLSTAG_H

#include "tset.h"
#include "stagfunc.h"

void SearchStaggers(int order, int k, StaggerFunc f, void *args,
                        boolean Opt);
        // Performs function f with arguments args upon every stagger of
        // the given order.
        //
        // If Opt is true, not all staggers are searched.
        // Instead, equivalences are taken into account.
        // Specifically, new values/rows/cols are only
        // taken in sequence.
        //
        // That is, val/row/col y will not be used before (y-1).
        //
        // Further equivalences involving sorting of entries within
        // rows are also taken into account.
```

```
#endif
```

A.8 Source for *allstag.cc*

```
// Implementation of functions described in AllStag.h.

#include "allstag.h"
#include "sinfo.h"

// For k-stagger of order n, requires n^2 iterations.

void search(SearchInfo &inf);
   // Function called for each individual step of the search.

void SearchStaggers(int order, int k, StaggerFunc f, void *args,
                boolean Opt)
{
TSet t(order);

if (Opt)
   search(OptSearchInfo(&t, k, order, f, args));
else
   search(SearchInfo(&t, k, order, f, args));
}

void search(SearchInfo &inf)
{
if (inf.r == inf.n)
   {
   (*(inf.f))(*inf.t, inf.args);
   return;
   }

// Can we leave this entry blank?
if (inf.CanLeaveBlank())
   {
   inf.Blanking();
   inf++;
   search(inf);
   inf--;
   inf.Blanked();
   }

if (! inf.CanUse())
   return;

int max = inf.MaxVal();

int i;
for (i=0; i <= max; i++)
   {
   if (inf.tot[0][i] == inf.k)
      continue;

   // Can use more of value i.
```

```
   Triple tr(i, inf.r, inf.c);
   if (inf.t->add(tr))
      {
      inf.Using(i);

      inf += tr;
      inf++;

      search(inf);

      inf--;
      inf -= tr;
      inf.t->remove(tr);

      inf.Used();
      }
   }
}
```

A.9 Source for *sinfo.h*

```
// -------
// SInfo.h
// -------
//
// Defines the SearchInfo object, which represents the current state
// of the search for staggers.  Such objects are used in backtracking.

#ifndef __SINFO_H
#define __SINFO_H

#include "intstack.h"
#include "tset.h"
#include "stagfunc.h"

// SearchInfo class: Defines a state of an ordinary search.

class SearchInfo
        {
        public:
                int r, c;
                int k, n;
                int *tot[3];
                   // Total numbers of entries for
                   // particular values, rows and columns.

        public:
                TSet *t;

                StaggerFunc f;
                void *args;

                SearchInfo(TSet *newt, int newk, int newn, StaggerFunc newf,
                                        void *newargs);
                virtual ~SearchInfo();
```

```
                void operator ++(int);
                void operator --(int);
                void operator +=(Triple &tr);
                void operator -=(Triple &tr);

                virtual boolean CanLeaveBlank();
                virtual boolean CanUse();

                virtual void Using(int) {}
                virtual void Used() {}
                virtual void Blanking() {}
                virtual void Blanked() {}

                virtual int MaxVal()
                   { return n-1; }
        };

// Class OptSearchInfo: Used in optimised searches.

class OptSearchInfo : public SearchInfo
        {
        public:
                int nextc, nextv;
                int firstc;

        protected:
                IntStack CStack, VStack, FCStack;

        public:
                OptSearchInfo(TSet *newt, int newk, int newn, StaggerFunc newf,
                                        void *newargs);

                virtual boolean CanLeaveBlank();
                virtual boolean CanUse();

                virtual void Using(int val);
                virtual void Used();
                virtual void Blanking();
                virtual void Blanked();

                virtual int MaxVal();
        };

#endif
```

A.10 Source for *sinfo.cc*

```
// Implements functions described in SInfo.h.

#include "sinfo.h"

SearchInfo::SearchInfo(TSet *newt, int newk, int newn, StaggerFunc newf,
                        void *newargs)
{
```

```
r = c = 0;
k = newk;
n = newn;

t = newt;

f = newf;
args = newargs;

int i, j;
for (i=0;i<3;i++)
   {
   tot[i] = new int[n];
   for (j=0; j<n; j++)
      tot[i][j] = 0;
   }
}

SearchInfo::~SearchInfo()
{
int i;
for (i=0;i<3;i++)
   delete[] tot[i];
}

void SearchInfo::operator ++(int)
{
c++;
if (c == n)
   {
   c = 0;
   r++;
   }
}

void SearchInfo::operator --(int)
{
if (c == 0)
   {
   c = n;
   r--;
   }
c--;
}

void SearchInfo::operator +=(Triple &tr)
{
int i;
for (i=0; i<3; i++)
   tot[i][tr[i]]++;
}

void SearchInfo::operator -=(Triple &tr)
{
int i;
for (i=0; i<3; i++)
```

```
   tot[i][tr[i]]--;
}

boolean SearchInfo::CanLeaveBlank()
{
// Implements derivable conditions
// that must be satisfied if a given cell is to
// be left blank.
return ((k - tot[1][r] < n - c) &&
                (k - tot[2][c] < n - r));
}

boolean SearchInfo::CanUse()
{
return ! ((tot[1][r] == k) || (tot[2][c] == k));
}

// -------------- OptSearchInfo ----------------

OptSearchInfo::OptSearchInfo(TSet *newt, int newk, int newn,
        StaggerFunc newf, void *newargs) :
        SearchInfo(newt, newk, newn, newf, newargs)
{
nextc = nextv = 0;
firstc = 0;
}

boolean OptSearchInfo::CanLeaveBlank()
{
if ((c == nextc) && (tot[1][r] < k))
   return False;

return SearchInfo::CanLeaveBlank();
}

boolean OptSearchInfo::CanUse()
{
if (c < firstc)
   return False;

return SearchInfo::CanUse();
}

void OptSearchInfo::Using(int val)
{
VStack.Push(nextv);
CStack.Push(nextc);
if (val == nextv)
   nextv++;
if (c == nextc)
   nextc++;
}

void OptSearchInfo::Used()
{
nextv = VStack.Pop();
```

```
nextc = CStack.Pop();
}

void OptSearchInfo::Blanking()
{
FCStack.Push(firstc);
if (c == firstc)
   firstc++;
}

void OptSearchInfo::Blanked()
{
firstc = FCStack.Pop();
}

int OptSearchInfo::MaxVal()
{
if (nextv == n)
   return n-1;
else
   return nextv;
}
```

A.11 Source for *intstack.h*

```
// ----------
// IntStack.h
// ----------
//
// Defines a stack of integers.

#ifndef __INTSTACK_H
#define __INTSTACK_H

// IntNode class: A member of an integer stack.

class IntNode
        {
        public:
                int val;
                IntNode *next;

        public:
                IntNode(int newval)
                   { val = newval; }
        };

// IntStack class: A stack of integers.

class IntStack
        {
        protected:
                IntNode *first;

        public:
```

```
                IntStack()
                   { first = 0; }
                virtual ~IntStack();

                void Push(int val);
                   // Push an integer onto the stack.
                int Pop();
                   // Pop an integer from the stack.
        };

#endif
```

A.12 Source for *intstack.cc*

```
// Implements functions described in IntStack.h.

#include "intstack.h"

IntStack::~IntStack()
{
while (first != 0)
   Pop();
}

void IntStack::Push(int val)
{
IntNode *p = new IntNode(val);
p->next = first;
first = p;
}

int IntStack::Pop()
{
if (first == 0)
   return 0;

IntNode *p = first;
first = first->next;
int ans = p->val;
delete p;
return ans;
}
```

A.13 Source for *stagger.cc*

```
// This is the main program.
// Values for k and n are read from standard input, and
//    all invalid k-staggers of order n are then written
//    to standard output.

#include "output.h"
#include "allstag.h"

void OutStagTSet(TSet &t, void *);
```

```
main()
{
int n, k;

cin >> k >> n;
while (n != 0)
   {
   cout << "\nInvalid " << k << "-Staggers of order " << n << " :\n\n";
   SearchStaggers(n, k, OutStagTSet, 0, True);
   cin >> k >> n;
   }
return 0;
}

void OutStagTSet(TSet &t, void *)
{
if (! t.valid())
   {
   outset(t, cout);
   cout << '\n';
   cout.flush();
   }
}
```