

Latin Square

Alexander Bentkamp

February 23, 2021

Abstract

A theory about Latin Squares following [1]. A Latin Square is a $n \times n$ table filled with integers from 1 to n where each number appears exactly once in each row and each column. A Latin Rectangle is a partially filled $n \times n$ table with r filled rows and $n - r$ empty rows, such that each number appears at most once in each row and each column. The main result of this theory is that any Latin Rectangle can be completed to a Latin Square.

Contents

```
theory Latin-Square  
imports Marriage.Marriage  
begin
```

This theory is about Latin Squares. A Latin Square is a $n \times n$ table filled with integers from 1 to n where each number appears exactly once in each row and each column.

As described in "Das Buch der Beweise" a nice way to describe these squares by a $3 \times n$ matrix. Each column of this matrix contains the index of the row r , the index of the column c and the number in the cell (r,c) . This $3 \times n$ matrix is called orthogonal array ("Zeilenmatrix").

I thought about different ways to formalize this orthogonal array, and came up with this: As the order of the columns in the array does not matter at all and no column can be a duplicate of another column, the orthogonal array is in fact a set of 3-tuples. Another advantage of formalizing it as a set is that it can easily model partially filled squares. For these 3-tuples I decided against 3-lists and against $nat \times nat \times nat$ (which is really $(nat \times nat) \times nat$) in favor of a function from a type with three elements to nat .

Additionally I use the numbers 0 to $n - 1$ instead of 1 to n for indexing the rows and columns as well as for filling the cells.

```
datatype latin-type = Row | Col | Num
```

latin_type is of sort enum, needed for "value" command

instantiation latin-type :: enum

begin

definition enum-latin-type == [Row, Col, Num]

definition enum-all-latin-type (P:: latin-type => bool) = (P Row & P Col & P Num)

definition enum-ex-latin-type (P:: latin-type => bool) = ($\exists x. P x$)

instance

apply standard

apply (auto simp add: enum-latin-type-def enum-all-latin-type-def enum-ex-latin-type-def)

apply (case-tac x, auto)

by (metis latin-type.exhaust)

end

Given a latin_type t, you might want to reference the other two. These are "next t" and "next (next t)":

definition [simp]:next t \equiv (case t of Row => Col | Col => Num | Num => Row)

lemma all-types-next-equiv:($\forall t. P (\text{next } t) \longleftrightarrow (\forall t. P t)$)

apply (rule iffI)

using next-def latin-type.case latin-type.exhaust **apply** metis

apply metis

done

We call a column of the orthogonal array a latin_entry:

type-synonym latin-entry = latin-type => nat

This function removes one element of the 3-tupel and returns the other two as a pair:

definition without :: latin-type => latin-entry => nat \times nat **where**

[simp]:without t $\equiv \lambda e. (e (\text{next } t), e (\text{next } (\text{next } t)))$

value without Row ($\lambda t. \text{case } t \text{ of Row} \Rightarrow 0 \mid \text{Col} \Rightarrow 1 \mid \text{Num} \Rightarrow 2$) — returns (1,2)

abbreviation row-col \equiv without Num

returns row and column of a latin_entry as a pair.

abbreviation col-num \equiv without Row

returns column and number of a latin_entry as a pair.

abbreviation num-row \equiv without Col

returns number and row of a latin_entry as a pair.

A partial latin square is a square that contains each number at most once in each row and each column, but not all cells have to be filled. Equivalently

we can say that any two rows of the orthogonal array contain each pair of two numbers at most once. This can be expressed using the `inj_on` predicate:

definition *partial-latin-square* :: *latin-entry set* \Rightarrow *nat* \Rightarrow *bool* **where**
partial-latin-square *s n* \equiv

($\forall t.$ *inj-on* (*without t*) *s*) \wedge — numbers are unique in each column ($t=$ Row),
 numbers are unique in each row ($t=$ Col), rows-column combinations are specified
 unambiguously ($t=$ Num)

($\forall e \in s. \forall t. e \ t < n$) — all numbers, column indices and row indices are $<n$

value *partial-latin-square* {
 ($\lambda t.$ *case t of Row* \Rightarrow 0 | *Col* \Rightarrow 1 | *Num* \Rightarrow 0),
 ($\lambda t.$ *case t of Row* \Rightarrow 1 | *Col* \Rightarrow 0 | *Num* \Rightarrow 1)
 } 2 — True

value *partial-latin-square* {
 ($\lambda t.$ *case t of Row* \Rightarrow 0 | *Col* \Rightarrow 0 | *Num* \Rightarrow 1),
 ($\lambda t.$ *case t of Row* \Rightarrow 1 | *Col* \Rightarrow 0 | *Num* \Rightarrow 1)
 } 2 — False, because 1 appears twice in column 0

Looking at the orthogonal array a latin square is given iff any two rows of the orthogonal array contain each pair of two numbers at exactly once:

definition *latin-square* :: *latin-entry set* \Rightarrow *nat* \Rightarrow *bool* **where**
latin-square *s n* \equiv

($\forall t.$ *bij-betw* (*without t*) *s* ($\{0..<n\} \times \{0..<n\}$))

value *latin-square* {
 ($\lambda t.$ *case t of Row* \Rightarrow 0 | *Col* \Rightarrow 0 | *Num* \Rightarrow 1), ($\lambda t.$ *case t of Row* \Rightarrow 0 | *Col* \Rightarrow
 1 | *Num* \Rightarrow 0),
 ($\lambda t.$ *case t of Row* \Rightarrow 1 | *Col* \Rightarrow 0 | *Num* \Rightarrow 0), ($\lambda t.$ *case t of Row* \Rightarrow 1 | *Col* \Rightarrow
 1 | *Num* \Rightarrow 1)
 } 2 — True

value *latin-square* {
 ($\lambda t.$ *case t of Row* \Rightarrow 0 | *Col* \Rightarrow 0 | *Num* \Rightarrow 1), ($\lambda t.$ *case t of Row* \Rightarrow 0 | *Col* \Rightarrow
 1 | *Num* \Rightarrow 0),
 ($\lambda t.$ *case t of Row* \Rightarrow 1 | *Col* \Rightarrow 0 | *Num* \Rightarrow 0), ($\lambda t.$ *case t of Row* \Rightarrow 1 | *Col* \Rightarrow
 1 | *Num* \Rightarrow 0)
 } 2 — False, because 0 appears twice in Col 1 and twice in Row 1

A latin rectangle is a partial latin square in which the first *m* rows are filled and the following rows are empty:

definition *latin-rect* :: *latin-entry set* \Rightarrow *nat* \Rightarrow *nat* \Rightarrow *bool* **where**
latin-rect *s m n* \equiv

$m \leq n \wedge$

partial-latin-square *s n* \wedge

bij-betw *row-col* *s* ($\{0..<m\} \times \{0..<n\}$) \wedge

bij-betw *num-row* *s* ($\{0..<n\} \times \{0..<m\}$)

value *latin-rect* {

```

  ( $\lambda t. \text{case } t \text{ of Row} \Rightarrow 0 \mid \text{Col} \Rightarrow 0 \mid \text{Num} \Rightarrow 1$ ), ( $\lambda t. \text{case } t \text{ of Row} \Rightarrow 0 \mid \text{Col} \Rightarrow 1 \mid \text{Num} \Rightarrow 0$ )
} 1 2 — True

```

```

value latin-rect {
  ( $\lambda t. \text{case } t \text{ of Row} \Rightarrow 0 \mid \text{Col} \Rightarrow 0 \mid \text{Num} \Rightarrow 1$ ), ( $\lambda t. \text{case } t \text{ of Row} \Rightarrow 0 \mid \text{Col} \Rightarrow 1 \mid \text{Num} \Rightarrow 0$ ),
  ( $\lambda t. \text{case } t \text{ of Row} \Rightarrow 1 \mid \text{Col} \Rightarrow 0 \mid \text{Num} \Rightarrow 0$ ), ( $\lambda t. \text{case } t \text{ of Row} \Rightarrow 1 \mid \text{Col} \Rightarrow 1 \mid \text{Num} \Rightarrow 1$ )
} 1 2 — False

```

There is another equivalent description of latin rectangles, which is easier to prove:

lemma *latin-rect-iff*:

$m \leq n \wedge \text{partial-latin-square } s \ n \wedge \text{card } s = n * m \wedge (\forall e \in s. e \text{ Row} < m) \longleftrightarrow \text{latin-rect } s \ m \ n$

proof (*rule iffI*)

assume *prems*: $m \leq n \wedge \text{partial-latin-square } s \ n \wedge \text{card } s = n * m \wedge (\forall e \in s. e \text{ Row} < m)$

have *bij1*: *bij-betw row-col s* ($\{0..<m\} \times \{0..<n\}$) **using** *prems*

proof

have *inj-on row-col s* **using** *prems partial-latin-square-def* **by** *blast*

moreover **have** $\{0..<m\} \times \{0..<n\} = \text{row-col } 's$

proof—

have $\text{row-col } 's \subseteq \{0..<m\} \times \{0..<n\}$ **using** *prems partial-latin-square-def* **by** *auto*

moreover **have** $\text{card } (\text{row-col } 's) = \text{card } (\{0..<m\} \times \{0..<n\})$ **using** *prems card-image[OF inj-on row-col s]* **by** *auto*

ultimately show $\{0..<m\} \times \{0..<n\} = \text{row-col } 's$ **using** *card-subset-eq[of {0..<m} × {0..<n} row-col 's]* **by** *auto*

qed

ultimately show *?thesis* **unfolding** *bij-betw-def* **by** *auto*

qed

have *bij2*: *bij-betw num-row s* ($\{0..<n\} \times \{0..<m\}$) **using** *prems*

proof

have *inj-on num-row s* **using** *prems partial-latin-square-def* **by** *blast*

moreover **have** $\{0..<n\} \times \{0..<m\} = \text{num-row } 's$

proof—

have $\text{num-row } 's \subseteq \{0..<n\} \times \{0..<m\}$ **using** *prems partial-latin-square-def* **by** *auto*

moreover **have** $\text{card } (\text{num-row } 's) = \text{card } (\{0..<n\} \times \{0..<m\})$ **using** *prems card-image[OF inj-on num-row s]* **by** *auto*

ultimately show $\{0..<n\} \times \{0..<m\} = \text{num-row } 's$ **using** *card-subset-eq[of {0..<n} × {0..<m} num-row 's]* **by** *auto*

qed

ultimately show *?thesis* **unfolding** *bij-betw-def* **by** *auto*

qed

```

from prems bij1 bij2 show latin-rect s m n unfolding latin-rect-def by auto
next
  assume prems:latin-rect s m n
  have  $m \leq n$  partial-latin-square s n using latin-rect-def prems by auto
  moreover have  $\text{card } s = m * n$ 
  proof –
    have bij-betw row-col s ( $\{0..<m\} \times \{0..<n\}$ ) using latin-rect-def prems by
auto
    then show ?thesis using bij-betw-same-card[of row-col s {0..<m} × {0..<n}]
by auto
  qed
  moreover have  $\forall e \in s. e \text{ Row} < m$  using latin-rect-def prems using atLeast0LessThan
bij-betwE by fastforce
  ultimately show  $m \leq n \wedge \text{partial-latin-square } s \ n \wedge \text{card } s = n * m \wedge (\forall e \in s. e$ 
Row < m) by auto
qed

```

A square is a latin square iff it is a partial latin square with all n^2 cells filled:

lemma *partial-latin-square-full*:

partial-latin-square s n \wedge $\text{card } s = n * n \iff \text{latin-square } s \ n$

proof (*rule iffI*)

assume *prem: partial-latin-square s n* \wedge $\text{card } s = n * n$

have $\forall t. (\text{without } t) \text{ ' } s \subseteq \{0..<n\} \times \{0..<n\}$

proof

fix *t* **show** (*without t*) $\text{' } s \subseteq \{0..<n\} \times \{0..<n\}$ **using** *partial-latin-square-def*
next-def atLeast0LessThan prem **by** (*cases t*) *auto*

qed

then **show** *partial-latin-square s n* \wedge $\text{card } s = n * n \implies \text{latin-square } s \ n$

unfolding *latin-square-def* **using** *partial-latin-square-def*

by (*metis bij-betw-def card-atLeastLessThan card-cartesian-product card-image*
card-subset-eq diff-zero finite-SigmaI finite-atLeastLessThan)

next

assume *prem:latin-square s n*

then **have** *bij-betw row-col s* ($\{0..<n\} \times \{0..<n\}$) **using** *latin-square-def* **by**
blast

moreover **have** *partial-latin-square s n*

proof –

have $\forall t. \forall e \in s. (\text{without } t) e \in (\{0..<n\} \times \{0..<n\})$ **using** *prem latin-square-def*
bij-betwE **by** *metis*

then **have** $1: \forall e \in s. \forall t. e \ t < n$ **using** *latin-square-def all-types-next-equiv[of*
 $\lambda t. \forall e \in s. e \ t < n]$ *bij-betwE* **by** *auto*

have $2: (\forall t. \text{inj-on } (\text{without } t) \ s)$ **using** *prem bij-betw-def latin-square-def* **by**
auto

from $1 \ 2$ **show** *?thesis* **using** *partial-latin-square-def* **by** *auto*

qed

ultimately **show** *partial-latin-square s n* \wedge $\text{card } s = n * n$ **by** (*auto simp add:*
bij-betw-same-card)

qed

Now we prove Lemma 1 from chapter 27 in "Das Buch der Beweise". But first some lemmas, that prove very intuitive facts:

lemma *bij-restrict*:

assumes *bij-betw* $f A B \forall a \in A. P a \longleftrightarrow Q (f a)$

shows *bij-betw* $f \{a \in A. P a\} \{b \in B. Q b\}$

proof –

have *inj*: *inj-on* $f \{a \in A. P a\}$ **using** *assms* *bij-betw-def* **by** (*metis* (*mono-tags*, *lifting*) *inj-onD* *inj-onI* *mem-Collect-eq*)

have *surj1*: $f ' \{a \in A. P a\} \subseteq \{b \in B. Q b\}$ **using** *assms*(1) *assms*(2) *bij-betwE* **by** *blast*

have *surj2*: $\{b \in B. Q b\} \subseteq f ' \{a \in A. P a\}$

proof

fix b

assume $b \in \{b \in B. Q b\}$

then obtain a **where** $f a = b$ $a \in A$ **using** *assms*(1) *bij-betw-inv-into-right* *bij-betwE* *bij-betw-inv-into* *mem-Collect-eq* **by** (*metis* (*no-types*, *lifting*))

then show $b \in f ' \{a \in A. P a\}$ **using** $\langle b \in \{b \in B. Q b\} \rangle$ *assms*(2) **by** *blast*

qed

with *inj* *surj1* *surj2* **show** *?thesis* **using** *bij-betw-imageI* **by** *fastforce*

qed

lemma *cartesian-product-margin1*:

assumes $a \in A$

shows $\{p \in A \times B. fst p = a\} = \{a\} \times B$

using *SigmaI* *assms* **by** *auto*

lemma *cartesian-product-margin2*:

assumes $b \in B$

shows $\{p \in A \times B. snd p = b\} = A \times \{b\}$

using *SigmaI* *assms* **by** *auto*

The union of sets containing at most k elements each cannot contain more elements than the number of sets times k :

lemma *limited-family-union*: *finite* $B \implies \forall P \in B. card P \leq k \implies card (\bigcup B) \leq card B * k$

proof (*induction* B *rule:finite-induct*)

case *empty*

then show *?case* **by** *auto*

next

case (*insert* $P B$)

have $card (\bigcup (\text{insert } P B)) \leq card P + card (\bigcup B)$ **by** (*simp* *add: card-Un-le*)

then have $card (\bigcup (\text{insert } P B)) \leq card P + card B * k$ **using** *insert* **by** *auto*

then show *?case* **using** *insert* **by** *simp*

qed

If f hits each element at most k times, the domain of f can only be k times bigger than the image of f :

lemma *limited-preimages*:
assumes $\forall x \in f^{-1} D. \text{card} ((f^{-1} \{x\}) \cap D) \leq k$ *finite D*
shows $\text{card } D \leq \text{card} (f^{-1} D) * k$
proof –
let $?preimages = (\lambda x. (f^{-1} \{x\}) \cap D)^{-1} (f^{-1} D)$
have $D = \bigcup ?preimages$ **by** *auto*
have $\text{card} (\bigcup ?preimages) \leq \text{card} ?preimages * k$ **using** *limited-family-union*[of $?preimages$ k] **assms** **by** *auto*
moreover **have** $\text{card} (?preimages) * k \leq \text{card} (f^{-1} D) * k$ **using** *card-image-le*[of $f^{-1} D$ $\lambda x. (f^{-1} \{x\}) \cap D$] **assms** **by** *auto*
ultimately **have** $\text{card} (\bigcup ?preimages) \leq \text{card} (f^{-1} D) * k$ **using** *le-trans* **by** *blast*
then show $?thesis$ **using** $\langle D = \bigcup ?preimages \rangle$ **by** *metis*
qed

Let A_1, \dots, A_n be sets with $k > 0$ elements each. Any element is only contained in at most k of these sets. Then there are more different elements in total than sets A_i :

lemma *union-limited-replicates*:
assumes *finite I* $\forall i \in I. \text{finite} (A\ i)$ $k > 0$ $\forall i \in I. \text{card} (A\ i) = k$ $\forall i \in I. \forall x \in (A\ i). \text{card} \{i \in I. x \in A\ i\} \leq k$
shows $\text{card} (\bigcup i \in I. (A\ i)) \geq \text{card } I$ **using** *assms*
proof –
let $?pairs = \{(i, x). i \in I \wedge x \in A\ i\}$

have *card-pairs*: $\text{card } ?pairs = \text{card } I * k$ **using** *assms*
proof (*induction I rule:finite-induct*)
case *empty*
then show *?case* **using** *card-eq-0-iff* **by** *auto*
next
case (*insert i0 I*)
have $\forall i \in I. \forall x \in (A\ i). \text{card} \{i \in I. x \in A\ i\} \leq k$
proof (*rule ballI*)
fix $i\ x$ **assume** $i \in I\ x \in A\ i$
then have $\text{card} \{i \in \text{insert } i0\ I. x \in A\ i\} \leq k$ **using** *insert* **by** *auto*
moreover **have** *finite* $\{i \in \text{insert } i0\ I. x \in A\ i\}$ **using** *insert* **by** *auto*
ultimately show $\text{card} \{i \in I. x \in A\ i\} \leq k$ **using** *card-mono*[of $\{i \in \text{insert } i0\ I. x \in A\ i\}$ $\{i \in I. x \in A\ i\}$] *le-trans* **by** *blast*
qed
then have *card-S*: $\text{card} \{(i, x). i \in I \wedge x \in A\ i\} = \text{card } I * k$ **using** *insert* **by** *auto*

have *card-B*: $\text{card} \{(i, x). i = i0 \wedge x \in A\ i0\} = k$ **using** *insert* **by** *auto*

have $\{(i, x). i \in \text{insert } i0\ I \wedge x \in A\ i\} = \{(i, x). i \in I \wedge x \in A\ i\} \cup \{(i, x). i = i0 \wedge x \in A\ i0\}$ **by** *auto*
moreover **have** $\{(i, x). i \in I \wedge x \in A\ i\} \cap \{(i, x). i = i0 \wedge x \in A\ i0\} = \{\}$
using *insert* **by** *auto*
moreover **have** *finite* $\{(i, x). i \in I \wedge x \in A\ i\}$ **using** *insert rev-finite-subset*[of $I \times \bigcup (A^{-1} T)$ $\{(i, x). i \in I \wedge x \in A\ i\}$] **by** *auto*

moreover have $\text{finite } \{(i, x). i=i0 \wedge x \in A \ i0\}$ **using** $\text{insert card-B card.infinite neq0-conv}$ **by** blast
ultimately have $\text{card } \{(i, x). i \in \text{insert } i0 \ I \wedge x \in A \ i\} = \text{card } \{(i, x). i \in I \wedge x \in A \ i\} + \text{card } \{(i, x). i=i0 \wedge x \in A \ i0\}$ **by** $(\text{simp add: card-Un-disjoint})$
with card-S card-B **have** $\text{card } \{(i, x). i \in \text{insert } i0 \ I \wedge x \in A \ i\} = (\text{card } I + 1) * k$ **by** auto
then show $?case$ **using** insert **by** auto
qed

define f **where** $f \ ix = (\text{case } ix \ \text{of } (i,x) \Rightarrow x)$ **for** $ix :: 'a \times 'b$

have $\text{preimages-le-k: } \forall x \in f^{-1} \ ?pairs. \text{card } ((f^{-1} \ \{x\}) \cap \ ?pairs) \leq k$
proof

fix $x0$ **assume** $x0\text{-def: } x0 \in f^{-1} \ ?pairs$

have $(f^{-1} \ \{x0\}) \cap \ ?pairs = \{(i,x). i \in I \wedge x \in A \ i \wedge x=x0\}$ **using** $f\text{-def}$ **by** auto
moreover have $\text{card } \{(i,x). i \in I \wedge x \in A \ i \wedge x=x0\} = \text{card } \{i \in I. x0 \in A \ i\}$

using $(\text{finite } I)$

proof –

have $\text{inj-on } (\lambda i. (i,x0)) \ \{i \in I. x0 \in A \ i\}$ **by** $(\text{meson Pair-inject inj-onI})$

moreover have $(\lambda i. (i,x0))^{-1} \ \{i \in I. x0 \in A \ i\} = \{(i,x). i \in I \wedge x \in A \ i \wedge x=x0\}$

by $(\text{rule subset-antisym})$ blast+

ultimately show $?thesis$ **using** card-image **by** fastforce

qed

ultimately have $1:\text{card } ((f^{-1} \ \{x0\}) \cap \ ?pairs) = \text{card } \{i \in I. x0 \in A \ i\}$ **by** auto

have $\exists i0. x0 \in A \ i0 \wedge i0 \in I$ **using** $x0\text{-def } f\text{-def}$ **by** auto

then have $\text{card } \{i \in I. x0 \in A \ i\} \leq k$ **using** assms **by** auto

with 1 **show** $\text{card } ((f^{-1} \ \{x0\}) \cap \ ?pairs) \leq k$ **by** auto

qed

have $\text{card } \ ?pairs \leq \text{card } (f^{-1} \ ?pairs) * k$

proof –

have $\text{finite } \{(i, x). i \in I \wedge x \in A \ i\}$ **using** $\text{assms card-pairs not-finite-existsD}$
by fastforce

then show $?thesis$ **using** $\text{limited-preimages[of } f \ ?pairs \ k, \ OF \ \text{preimages-le-k}]$
by auto

qed

then have $\text{card } I \leq \text{card } (f^{-1} \ ?pairs)$ **using** card-pairs assms **by** auto

moreover have $f^{-1} \ ?pairs = (\bigcup i \in I. (A \ i))$ **using** $f\text{-def } [abs\text{-def}]$ **by** auto

ultimately show $?thesis$ **using** $f\text{-def}$ **by** auto

qed

In a $m \times n$ latin rectangle each number appears in m columns:

lemma $\text{latin-rect-card-col:}$

assumes $\text{latin-rect } s \ m \ n \ x < n$

shows $\text{card } \{e \ \text{Col} \mid e. e \in s \wedge e \ \text{Num} = x\} = m$

proof –

have $\text{card } \{e \in s. e \ \text{Num} = x\} = m$

proof –
have 1:*bij-betw num-row s* ($\{0..<n\} \times \{0..<m\}$) **using** *assms latin-rect-def* **by** *auto*
have 2: $\forall e \in s. e \text{ Num} = x \longleftrightarrow \text{fst} (\text{num-row } e) = x$ **by** *simp*
have *bij-betw num-row* $\{e \in s. e \text{ Num} = x\}$ ($\{x\} \times \{0..<m\}$)
using *bij-restrict[OF 1 2] cartesian-product-margin1*[of $x \{0..<n\} \{0..<m\}$]
assms **by** *auto*
then show *?thesis* **using** *card-cartesian-product* **by** (*simp add: bij-betw-same-card*)
qed
moreover have $\text{card } \{e \in s. e \text{ Num} = x\} = \text{card } \{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\}$
proof –
have *inj-on col-num s* **using** *assms latin-rect-def*[of $s \ m \ n$] *partial-latin-square-def*[of $s \ n$] **by** *blast*
then have *inj-on col-num* $\{e \in s. e \text{ Num} = x\}$ **by** (*metis (mono-tags, lifting) inj-onD inj-onI mem-Collect-eq*)
then have *inj-on* $(\lambda e. e \text{ Col}) \{e \in s. e \text{ Num} = x\}$ **unfolding** *inj-on-def* **using** *without-def* **by** *auto*
moreover have $(\lambda e. e \text{ Col}) ' \{e \in s. e \text{ Num} = x\} = \{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\}$ **by** (*rule subset-antisym*) *blast+*
ultimately show *?thesis* **using** *card-image* **by** *fastforce*
qed
ultimately show *?thesis* **by** *auto*
qed

In a $m \times n$ latin rectangle each column contains m numbers:

lemma *latin-rect-card-num*:
assumes *latin-rect s m n x < n*
shows $\text{card } \{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = x\} = m$
proof –
have $\text{card } \{e \in s. e \text{ Col} = x\} = m$
proof –
have 1:*bij-betw row-col s* ($\{0..<m\} \times \{0..<n\}$) **using** *assms latin-rect-def* **by** *auto*
have 2: $\forall e \in s. e \text{ Col} = x \longleftrightarrow \text{snd} (\text{row-col } e) = x$ **by** *simp*
have *bij-betw row-col* $\{e \in s. e \text{ Col} = x\}$ ($\{0..<m\} \times \{x\}$)
using *bij-restrict[OF 1 2] cartesian-product-margin2*[of $x \{0..<n\} \{0..<m\}$]
assms **by** *auto*
then show *?thesis* **using** *card-cartesian-product* **by** (*simp add: bij-betw-same-card*)
qed
moreover have $\text{card } \{e \in s. e \text{ Col} = x\} = \text{card } \{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = x\}$
proof –
have *inj-on col-num s* **using** *assms latin-rect-def*[of $s \ m \ n$] *partial-latin-square-def*[of $s \ n$] **by** *blast*
then have *inj-on col-num* $\{e \in s. e \text{ Col} = x\}$ **by** (*metis (mono-tags, lifting) inj-onD inj-onI mem-Collect-eq*)
then have *inj-on* $(\lambda e. e \text{ Num}) \{e \in s. e \text{ Col} = x\}$ **unfolding** *inj-on-def* **using** *without-def* **by** *auto*
moreover have $(\lambda e. e \text{ Num}) ' \{e \in s. e \text{ Col} = x\} = \{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = x\}$ **by** (*rule subset-antisym*) *blast+*

ultimately show *?thesis* **using** *card-image* **by** *fastforce*
qed
ultimately show *?thesis* **by** *auto*
qed

Finally we prove lemma 1 chapter 27 of "Das Buch der Beweise":

theorem
assumes *latin-rect* s $(n-m)$ n $m \leq n$
shows $\exists s'. s \subseteq s' \wedge$ *latin-square* $s' n$
using *assms*
proof (*induction* m *arbitrary:s*) — induction over the number of empty rows
case 0
then have *bij-betw* *row-col* s $(\{0..<n\} \times \{0..<n\})$ **using** *latin-rect-def* **by** *auto*
then have *card* $s = n * n$ **by** (*simp* *add:bij-betw-same-card*)
then show *?case* **using** *partial-latin-square-full* 0 *latin-rect-def* **by** *auto*
next
case (*Suc* m)

— We use the Hall theorem on the sets A_j of numbers that do not occur in column j :
let *?not-in-column* $= \lambda j. \{0..<n\} - \{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = j\}$

— Proof of the hall condition:
have $\forall J \subseteq \{0..<n\}. \text{card } J \leq \text{card } (\bigcup_{j \in J}. \text{?not-in-column } j)$
proof (*rule* *allI*; *rule* *impI*)
fix J **assume** $J\text{-def}: J \subseteq \{0..<n\}$
have $\forall j \in J. \text{card } (\text{?not-in-column } j) = \text{Suc } m$
proof
fix j **assume** $j\text{-def}: j \in J$
have $\{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = j\} \subseteq \{0..<n\}$ **using** *atLeastLessThan-iff* *Suc*
latin-rect-def *partial-latin-square-def* **by** *auto*
moreover then have *finite* $\{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = j\}$ **using** *finite-subset*
by *auto*
ultimately have $\text{card } (\text{?not-in-column } j) = \text{card } \{0..<n\} - \text{card } \{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = j\}$ **using** *card-Diff-subset*[of $\{e \text{ Num} \mid e. e \in s \wedge e \text{ Col} = j\}$ $\{0..<n\}$] **by** *auto*
then show $\text{card}(\text{?not-in-column } j) = \text{Suc } m$ **using** *latin-rect-card-num* $J\text{-def}$
 $j\text{-def}$ *Suc* **by** *auto*
qed
moreover have $\forall j_0 \in J. \forall x \in \text{?not-in-column } j_0. \text{card } \{j \in J. x \in \text{?not-in-column } j\} \leq \text{Suc } m$
proof (*rule* *ballI*; *rule* *ballI*)
fix $j_0 x$ **assume** $j_0 \in J$ $x \in \text{?not-in-column } j_0$
then have $\text{card } (\{0..<n\} - \{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\}) = \text{Suc } m$
proof —
have $\text{card } \{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\} = n - \text{Suc } m$ **using** *latin-rect-card-col*
 $(x \in \text{?not-in-column } j_0)$ *Suc* **by** *auto*
moreover have $\{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\} \subseteq \{0..<n\}$ **using** *Suc* *latin-rect-def*
partial-latin-square-def **by** *auto*

moreover then have $\text{finite } \{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\}$ **using** *finite-subset*
by *auto*
ultimately show *?thesis* **using** *card-Diff-subset*[of $\{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\} \{0..<n\}$] **using** *Suc.prem*s **by** *auto*
qed
moreover have $\{j \in J. x \in \text{?not-in-column } j\} \subseteq \{0..<n\} - \{e \text{ Col} \mid e. e \in s \wedge e \text{ Num} = x\}$ **using** *Diff-mono* *J-def* **using** $\langle x \in \text{?not-in-column } j \rangle$ **by** *blast*
ultimately show $\text{card } \{j \in J. x \in \text{?not-in-column } j\} \leq \text{Suc } m$ **by** (*metis* (*no-types*, *lifting*) *card-mono* *finite-Diff* *finite-atLeastLessThan*)
qed
moreover have *finite* *J* **using** *J-def* *finite-subset* **by** *auto*
ultimately show $\text{card } J \leq \text{card } (\bigcup_{j \in J. \text{?not-in-column } j})$ **using** *union-limited-replicates*[of *J* *?not-in-column* *Suc m*] **by** *auto*
qed

— The Hall theorem gives us a system of distinct representatives, which we can use to fill the next row:

then obtain *R* **where** *R-def*: $\forall j \in \{0..<n\}. R \ j \in \text{?not-in-column } j \wedge \text{inj-on } R \ \{0..<n\}$ **using** *marriage-HV*[of $\{0..<n\}$ *?not-in-column*] **by** *blast*

define *new-row* **where** $\text{new-row} = (\lambda j. \text{rec-latin-type } (n - \text{Suc } m) \ j \ (R \ j)) \ \{0..<n\}$

define *s'* **where** $s' = s \cup \text{new-row}$

— *s'* is now a latin rect with one more row:

have *latin-rect* *s'* $(n-m) \ n$

proof —

— We prove all four criteria specified in the lemma *latinrectiff*:

have $n-m \leq n$ **by** *auto*

moreover have *partial-latin-square* *s'* *n*

proof —

have *inj-on* (*without Col*) *s'* **unfolding** *inj-on-def*

proof (*rule ballI*; *rule ballI*; *rule impI*)

fix *e1 e2* **assume** $e1 \in s' \ e2 \in s' \ \text{num-row } e1 = \text{num-row } e2$

then have $e1 \text{ Num} = e2 \text{ Num} \ e1 \text{ Row} = e2 \text{ Row}$ **using** *without-def* **by** *auto*

moreover have $e1 \text{ Col} = e2 \text{ Col}$

proof (*cases*)

assume $e1 \text{ Row} = n - \text{Suc } m$

then have $e2 \text{ Row} = n - \text{Suc } m$ **using** *without-def* $\langle \text{num-row } e1 = \text{num-row } e2 \rangle$ **by** *auto*

have $\forall e \in s. e \text{ Row} < n - \text{Suc } m$ **using** *Suc latin-rect-iff* **by** *blast*

then have $e1 \in \text{new-row} \ e2 \in \text{new-row}$ **using** *s'-def* $\langle e1 \in s' \rangle \langle e2 \in s' \rangle$ $\langle e1 \text{ Row} = n - \text{Suc } m \rangle \langle e2 \text{ Row} = n - \text{Suc } m \rangle$ **by** *auto*

then have $e1 \text{ Num} = R \ (e1 \text{ Col}) \ e2 \text{ Num} = R \ (e2 \text{ Col})$ **using** *new-row-def* **by** *auto*

then have $R \ (e1 \text{ Col}) = R \ (e2 \text{ Col})$ **using** $\langle e1 \text{ Num} = e2 \text{ Num} \rangle$ **by** *auto*

moreover have $e1 \text{ Col} < n \ e2 \text{ Col} < n$ **using** $\langle e1 \in \text{new-row} \rangle \langle e2 \in \text{new-row} \rangle$ *new-row-def* **by** *auto*

ultimately show $e1 \text{ Col} = e2 \text{ Col}$ **using** *R-def* *inj-on-def* **by** (*metis*

(*mono-tags, lifting*) *atLeast0LessThan lessThan-iff*)

next

assume $e1 \text{ Row} \neq n - \text{Suc } m$

then have $e1 \in s \ e2 \in s$ **using** *new-row-def s'-def* $\langle e1 \in s' \rangle \langle e2 \in s' \rangle \langle e1 \text{ Row} = e2 \text{ Row} \rangle$ **by auto**

then show $e1 \text{ Col} = e2 \text{ Col}$ **using** *Suc latin-rect-def bij-betw-def* **by** (*metis* $\langle \text{num-row } e1 = \text{num-row } e2 \rangle \text{inj-onD}$)

qed

ultimately show $e1=e2$ **using** *latin-type.induct*[*of* $\lambda t. e1 \ t = e2 \ t$] **by auto**

qed

moreover have *inj-on* (*without Row*) *s'* **unfolding** *inj-on-def*

proof (*rule ballI; rule ballI; rule impI*)

fix $e1 \ e2$ **assume** $e1 \in s' \ e2 \in s' \ \text{col-num } e1 = \text{col-num } e2$

then have $e1 \text{ Col} = e2 \text{ Col} \ e1 \text{ Num} = e2 \text{ Num}$ **using** *without-def* **by auto**

moreover have $e1 \text{ Row} = e2 \text{ Row}$

proof (*cases*)

assume $e1 \text{ Row} = n - \text{Suc } m$

have $\forall e \in s. e \text{ Row} < n - \text{Suc } m$ **using** *Suc latin-rect-iff* **by blast**

then have $e2 \text{ Num} \in \text{?not-in-column } (e2 \text{ Col})$ **using** *R-def new-row-def* $\langle e1 \text{ Col} = e2 \text{ Col} \rangle \langle e1 \text{ Num} = e2 \text{ Num} \rangle$ **using** *s'-def* $\langle e1 \in s' \rangle \langle e1 \text{ Row} = n - \text{Suc } m \rangle$ **by auto**

then show $e1 \text{ Row} = e2 \text{ Row}$ **using** *new-row-def* $\langle e1 \text{ Row} = n - \text{Suc } m \rangle$ *s'-def* $\langle e2 \in s' \rangle$ **by auto**

next

assume $e1 \text{ Row} \neq n - \text{Suc } m$

then have $e1 \in s$ **using** *new-row-def s'-def* $\langle e1 \in s' \rangle$ **by auto**

then have $e2 \text{ Num} \notin \text{?not-in-column } (e2 \text{ Col})$ **using** $\langle e1 \text{ Col} = e2 \text{ Col} \rangle \langle e1 \text{ Num} = e2 \text{ Num} \rangle$ **by auto**

then have $e2 \in s$ **using** *new-row-def s'-def* $\langle e2 \in s' \rangle$ *R-def* **by auto**

moreover have *inj-on col-num s* **using** *Suc.premis latin-rect-def*[*of* $s \ (n - \text{Suc } m) \ n$] *partial-latin-square-def*[*of* $s \ n$] **by blast**

ultimately show $e1 \text{ Row} = e2 \text{ Row}$ **using** *Suc latin-rect-def* **by** (*metis* $\langle \text{col-num } e1 = \text{col-num } e2 \rangle \langle e1 \in s \rangle \text{inj-onD}$)

qed

ultimately show $e1=e2$ **using** *latin-type.induct*[*of* $\lambda t. e1 \ t = e2 \ t$] **by auto**

qed

moreover have *inj-on* (*without Num*) *s'* **unfolding** *inj-on-def*

proof (*rule ballI; rule ballI; rule impI*)

fix $e1 \ e2$ **assume** $e1 \in s' \ e2 \in s' \ \text{row-col } e1 = \text{row-col } e2$

then have $e1 \text{ Row} = e2 \text{ Row} \ e1 \text{ Col} = e2 \text{ Col}$ **using** *without-def* **by auto**

moreover have $e1 \text{ Num} = e2 \text{ Num}$

proof (*cases*)

assume $e1 \text{ Row} = n - \text{Suc } m$

then have $e2 \text{ Row} = n - \text{Suc } m$ **using** *without-def* $\langle \text{row-col } e1 = \text{row-col } e2 \rangle$ **by auto**

have $\forall e \in s. e \text{ Row} < n - \text{Suc } m$ **using** *Suc latin-rect-iff* **by blast**

then show $e1 \text{ Num} = e2 \text{ Num}$ **using** $\langle e1 \text{ Col} = e2 \text{ Col} \rangle$ **using** *new-row-def s'-def* $\langle e1 \in s' \rangle \langle e2 \in s' \rangle \langle e1 \text{ Row} = n - \text{Suc } m \rangle \langle e2 \text{ Row} = n - \text{Suc } m \rangle$ **by auto**

next

assume $e1 \text{ Row} \neq n - \text{Suc } m$
then have $e1 \in s \ e2 \in s$ **using** *new-row-def s'-def* $\langle e1 \in s' \rangle \langle e2 \in s' \rangle \langle e1 \text{ Row} = e2 \text{ Row} \rangle$ **by** *auto*
then show $e1 \text{ Num} = e2 \text{ Num}$ **using** *Suc latin-rect-def bij-betw-def* **by** $(\text{metis } \langle \text{row-col } e1 = \text{row-col } e2 \rangle \text{ inj-on } D)$
qed
ultimately show $e1 = e2$ **using** *latin-type.induct*[of $\lambda t. e1 \ t = e2 \ t$] **by** *auto*
qed
moreover have $\forall e \in s'. \forall t. e \ t < n$
proof (*rule ballI; rule allI*)
fix $e \ t$ **assume** $e \in s'$
then show $e \ t < n$
proof (*cases*)
assume $e \in \text{new-row}$
then show *?thesis* **using** *new-row-def R-def* **by** (*induction t*) *auto*
next
assume $e \notin \text{new-row}$
then show *?thesis* **using** *s'-def* $\langle e \in s' \rangle$ *latin-rect-def partial-latin-square-def*
Suc **by** *auto*
qed
qed
ultimately show *partial-latin-square s' n unfolding partial-latin-square-def*
using *latin-type.induct*[of $\lambda t. \text{inj-on } (\text{without } t) \ s'$] **by** *auto*
qed
moreover have $\text{card } s' = n * (n - m)$
proof –
have *card-s: card s = n * (n - Suc m)* **using** *latin-rect-iff Suc* **by** *auto*
have *card-new-row: card new-row = n* **unfolding** *new-row-def*
proof –
have *inj-on* $(\lambda j. \text{rec-latin-type } (n - \text{Suc } m) \ j \ (R \ j)) \ \{0..<n\}$ **unfolding**
inj-on-def
proof (*rule ballI; rule ballI; rule impI*)
fix $j1 \ j2$ **assume** $j1 \in \{0..<n\} \ j2 \in \{0..<n\}$ *rec-latin-type* $(n - \text{Suc } m)$
 $j1 \ (R \ j1) = \text{rec-latin-type } (n - \text{Suc } m) \ j2 \ (R \ j2)$
then show $j1 = j2$ **using** *latin-type.rec(2)*[of $(n - \text{Suc } m) \ j1 \ R \ j1$]
latin-type.rec(2)[of $- \ j2 \ -$] **by** *auto*
qed
then show $\text{card } ((\lambda j. \text{rec-latin-type } (n - \text{Suc } m) \ j \ (R \ j)) \ ' \ \{0..<n\}) = n$
by (*simp add: card-image*)
qed
have $s \cap \text{new-row} = \{\}$
proof –
have $\forall e \in s. e \ \text{Row} < n - \text{Suc } m$ **using** *Suc latin-rect-iff* **by** *blast*
then have $\forall e \in \text{new-row}. e \notin s$ **using** *new-row-def* **by** *auto*
then show *?thesis* **by** *blast*
qed
moreover have *finite s* **using** *Suc latin-rect-def* **by** (*metis bij-betw-finite finite-SigmaI finite-atLeastLessThan*)
moreover have *finite new-row* **using** *new-row-def* **by** *simp*

ultimately have $\text{card } s' = \text{card } s + \text{card } \text{new-row}$ **using** s' -def *card-Un-disjoint*
by *auto*
with *card-s card-new-row* **show** *?thesis* **using** *Suc* **by** (*metis Suc-diff-Suc*
Suc-le-lessD add.commute mult-Suc-right)
qed
moreover have $\forall e \in s'. e \text{ Row} < (n - m)$
proof (*rule ballI; cases*)
fix e
assume $e \in \text{new-row}$
then show $e \text{ Row} < n - m$ **using** *Suc new-row-def R-def* **by** *auto*
next
fix e
assume $e \in s' \ e \notin \text{new-row}$
then have $e \text{ Row} < n - \text{Suc } m$ **using** *latin-rect-iff Suc s'-def (e ∈ s')* **by**
auto
then show $e \text{ Row} < n - m$ **by** *auto*
qed
ultimately show *?thesis* **using** *latin-rect-iff[of n-m n]* **by** *auto*
qed

— Finally we use the induction hypothesis:
then obtain s'' **where** $s' \subseteq s''$ *latin-square s'' n* **using** *Suc* **by** *auto*
then have $s \subseteq s''$ **using** s' -def **by** *auto*
then show $\exists s'. s \subseteq s' \wedge \text{latin-square } s' n$ **using** (*latin-square s'' n*) **by** *auto*
qed

end

References

- [1] M. Aigner and G. Ziegler. *Das Buch der Beweise*. Springer, 2004.