# Latin Square 

Alexander Bentkamp

March 11, 2024


#### 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 <br> imports Marriage.Marriage <br> 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 ( $\mathrm{r}, \mathrm{c}$ ). 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 $\times$ $n a t) \times n a t$ ) 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 $\mid$ Col $\mid$ 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 }=>\mathrm{ bool ) = (P Row }\wedgePCol ^ P
Num)
    definition enum-ex-latin-type (P:: latin-type }=>\mathrm{ bool )}=(\existsx.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 $\Rightarrow \mathrm{Col}|\mathrm{Col} \Rightarrow \mathrm{Num}| \mathrm{Num} \Rightarrow$ Row $)$
lemma all-types-next-eqiv: $(\forall t . P($ 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 $\Rightarrow$ nat
This function removes one element of the 3 -tupel and returns the other two as a pair:
definition without $::$ latin-type $\Rightarrow$ latin-entry $\Rightarrow$ nat $\times$ nat where $[\operatorname{simp}]:$ without $t \equiv \lambda e .(e(n e x t), e(n e x t(n e x t ~ t)))$
value without Row ( $\lambda$ t. case $t$ of Row $\Rightarrow 0|C o l \Rightarrow 1| N u m \Rightarrow 2)$ - 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 ( $\mathrm{t}=\mathrm{Row}$ ), numbers are unique in each row ( $\mathrm{t}=\mathrm{Col}$ ), rows-column combinations are specified unambiguously ( $\mathrm{t}=\mathrm{Num}$ )
$(\forall e \in s . \forall t . e t<n)$ - all numbers, column indices and row indices are $<\mathrm{n}$
value partial-latin-square \{
( $\lambda$ t. case $t$ of Row $\Rightarrow 0|\operatorname{Col} \Rightarrow 1| N u m \Rightarrow 0)$,
( $\lambda$ t. case $t$ of Row $\Rightarrow 1|\mathrm{Col} \Rightarrow 0| \mathrm{Num} \Rightarrow 1$ )
\} 2 - True
value partial-latin-square \{
( $\lambda$ t. case $t$ of Row $\Rightarrow 0|\mathrm{Col} \Rightarrow 0| N u m \Rightarrow 1$ ),
$(\lambda t$. case $t$ of Row $\Rightarrow 1|\mathrm{Col} \Rightarrow 0| N u m \Rightarrow 1)$
\} 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 $\Rightarrow 0|\mathrm{Col} \Rightarrow 0| N u m \Rightarrow 1),(\lambda t$. case $t$ of Row $\Rightarrow 0 \mid \mathrm{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 0)$,
$(\lambda t$. case $t$ of Row $\Rightarrow 1|\mathrm{Col} \Rightarrow 0| \mathrm{Num} \Rightarrow 0),(\lambda t$. case $t$ of Row $\Rightarrow 1 \mid \mathrm{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 1)$
\} 2 - True
value latin-square \{
$(\lambda t$. case $t$ of Row $\Rightarrow 0|\mathrm{Col} \Rightarrow 0| \mathrm{Num} \Rightarrow 1),(\lambda t$. case $t$ of Row $\Rightarrow 0 \mid \mathrm{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 0)$,
$(\lambda t$. case $t$ of Row $\Rightarrow 1|\mathrm{Col} \Rightarrow 0| \mathrm{Num} \Rightarrow 0), \quad(\lambda t$. case $t$ of Row $\Rightarrow 1 \mid \mathrm{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 0)$
\} 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. case $t$ of Row $\Rightarrow 0|\mathrm{Col} \Rightarrow 0| \mathrm{Num} \Rightarrow 1), \quad(\lambda t$. case $t$ of Row $\Rightarrow 0 \mid \mathrm{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 0)$
\} 12 - True
value latin-rect \{
$(\lambda t$. case $t$ of Row $\Rightarrow 0|\mathrm{Col} \Rightarrow 0| \mathrm{Num} \Rightarrow 1), \quad(\lambda t$. case $t$ of Row $\Rightarrow 0 \mid \mathrm{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 0)$,
$(\lambda t$. case $t$ of Row $\Rightarrow 1|\mathrm{Col} \Rightarrow 0| N u m \Rightarrow 0),(\lambda t$. case $t$ of Row $\Rightarrow 1 \mid \operatorname{Col}$ $\Rightarrow 1 \mid N u m \Rightarrow 1)$
\} 12 - False
There is another equivalent description of latin rectangles, which is easier to prove:
lemma latin-rect-iff:
$m \leq n \wedge$ partial-latin-square $s n \wedge$ card $s=n * m \wedge(\forall e \in s . e$ Row $<m) \longleftrightarrow$ latin-rect $s m n$
proof (rule iffI)
assume prems: $m \leq n \wedge$ partial-latin-square $s n \wedge$ card $s=n * m \wedge(\forall e \in s$. e 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 $\{0 . .<m\} \times\{0 . .<n\}=$ row-col's
proof -
have row-col' $s \subseteq\{0 . .<m\} \times\{0 . .<n\}$ using prems partial-latin-square-def by auto
moreover have $\operatorname{card}($ row-col's $)=\operatorname{card}(\{0 . .<m\} \times\{0 . .<n\})$ using prems card-image $[O F\langle i n j$-on row-col s〉] by auto
ultimately show $\{0 . .<m\} \times\{0 . .<n\}=$ row-col's using card-subset-eq[of $\{0 . .<m\} \times\{0 . .<n\}$ 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 $\{0 . .<n\} \times\{0 . .<m\}=$ num-row' $s$
proof -
have num-row' $s \subseteq\{0 . .<n\} \times\{0 . .<m\}$ using prems partial-latin-square-def by auto
moreover have card (num-row's) $=\operatorname{card}(\{0 . .<n\} \times\{0 . .<m\})$ using prems card-image $[O F\langle i n j$-on num-row $s\rangle$ ] by auto
ultimately show $\{0 . .<n\} \times\{0 . .<m\}=$ num-row's using card-subset-eq[of $\{0 . .<n\} \times\{0 . .<m\}$ 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 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 $\{0 . .<m\} \times\{0 . .<n\}]$
by auto
qed
moreover have $\forall e \in s . e$ Row $<m$ using latin-rect-def prems using atLeastOLessThan bij-betwE by fastforce
ultimately show $m \leq n \wedge$ partial-latin-square s $n \wedge \operatorname{card} s=n * m \wedge(\forall e \in s . e$ 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 $n \wedge$ card $s=n * n \longleftrightarrow$ latin-square s $n$
proof (rule iffI)
assume prem: partial-latin-square s $n \wedge$ card $s=n * n$
have $\forall t$. (without $t$ )' $s \subseteq\{0 . .<n\} \times\{0 . .<n\}$
proof
fix $t$ show (without $t$ )' $s \subseteq\{0 . .<n\} \times\{0 . .<n\}$ using partial-latin-square-def next-def atLeastOLessThan prem by (cases $t$ ) auto
qed
then show partial-latin-square s $n \wedge$ card $s=n * n \Longrightarrow$ 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 $\forall t . \forall e \in s$. (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-eqiv[of $\lambda t$. $\forall e \in s$. e $t<n]$ bij-betwE by auto
have 2: $(\forall$ t. inj-on (without $t$ ) s) using prem bij-betw-def latin-square-def by auto
from 12 show ?thesis using partial-latin-square-def by auto
qed
ultimately show partial-latin-square $s n \wedge$ 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 \foralla\inA.P a\longleftrightarrowQ Q (f a)
shows bij-betw f {a\inA.Pa} {b\inB.Q b}
proof -
    have inj: inj-on f {a\inA.Pa} using assms bij-betw-def by (metis (mono-tags,
lifting) inj-onD inj-onI mem-Collect-eq)
    have surj1: f' {a\inA.Pa}\subseteq{b\inB.Q b} using assms(1) assms(2) bij-betwE
by blast
    have surj2: {b\inB.Q b}\subseteqf'{a\inA.Pa}
    proof
            fix b
            assume b f {b\inB.Qb}
            then obtain a where fa=b a\inA 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 f f'{a\inA.Pa} using <b\in{b\inB.Q b}> 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 $B \Longrightarrow \forall P \in B$. card $P \leq k \Longrightarrow$ 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 $(\bigcup($ insert $P B)) \leq$ card $P+$ card $(\bigcup B)$ by (simp add: card-Un-le)
then have card $(\bigcup($ 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 \(\forall x \in f^{\prime} D\). card \(\left(\left(f-{ }^{\prime}\{x\}\right) \cap D\right) \leq k\) finite \(D\)
shows card \(D \leq \operatorname{card}\left(f^{\prime} D\right) * k\)
proof -
    let ?preimages \(=(\lambda x .(f-‘\{x\}) \cap D)^{\prime}(f\) ‘ \(D)\)
    have \(D=\bigcup\) ? preimages by auto
    have card \((\bigcup\) ?preimages \() \leq\) card ?preimages \(* k\) using limited-family-union[of
?preimages \(k\) ] assms by auto
    moreover have card (?preimages) \(* k \leq \operatorname{card}\left(f^{\prime} D\right) * k\) using card-image-le[of
\(\left.f^{\prime} D \lambda x .(f-‘\{x\}) \cap D\right]\) assms by auto
    ultimately have card \((\bigcup\) ?preimages \() \leq \operatorname{card}\left(f^{\prime} D\right) * k\) using le-trans by blast
    then show ?thesis using \(\langle D=\bigcup\) ?preimages by metis
qed
```

Let $A_{1}, \ldots, A_{n}$ 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 $I \forall i \in I$. finite $(A i) k>0 \forall i \in I$. card $(A i)=k \forall i \in I . \forall x \in(A i)$. card $\{i \in I . x \in A i\} \leq k$
shows card $(\bigcup i \in I$. ( $A$ i $)$ ) $\geq$ card $I$ using assms
proof -
let ?pairs $=\{(i, x) . i \in I \wedge x \in A i\}$
have card-pairs: card ?pairs $=$ 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 $\forall i \in I . \forall x \in(A i)$. 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 card $\{i \in$ insert i0 I. $x \in A i\} \leq k$ using insert by auto
moreover have finite $\{i \in$ insert i0 $I . x \in A i\}$ using insert by auto
ultimately show card $\{i \in I . x \in A i\} \leq k$ using card-mono $[$ of $\{i \in$ insert io I. $x \in A i\}\{i \in I . x \in A i\}]$ le-trans by blast
qed
then have card-S: card $\{(i, x) . i \in I \wedge x \in A i\}=\operatorname{card} I * k$ using insert by auto
have card-B: card $\{(i, x) . i=i 0 \wedge x \in A i 0\}=k$ using insert by auto
have $\{(i, x) . i \in$ insert $i 0 I \wedge x \in A i\}=\{(i, x) . i \in I \wedge x \in A i\} \cup\{(i, x)$. $i=i 0 \wedge x \in A i 0\}$ by auto
moreover have $\{(i, x) . i \in I \wedge x \in A i\} \cap\{(i, x) . i=i 0 \wedge x \in A i 0\}=\{ \}$ using insert by auto
moreover have finite $\{(i, x) . i \in I \wedge x \in A i\}$ using insert rev-finite-subset[of $\left.I \times \bigcup\left(A^{\prime} I\right)\{(i, x) . i \in I \wedge x \in A i\}\right]$ by auto
moreover have finite $\{(i, x) . i=i 0 \wedge x \in A$ i0\} using insert card- $B$ card.infinite neq0-conv by blast
ultimately have $\operatorname{card}\{(i, x) . i \in$ insert i0 $I \wedge x \in A i\}=\operatorname{card}\{(i, x) . i \in I$ $\wedge x \in A i\}+$ card $\{(i, x) . i=i 0 \wedge x \in A i 0\}$ by (simp add: card-Un-disjoint)
with card-S card-B have card $\{(i, x) . i \in$ insert $i 0 I \wedge x \in A i\}=(\operatorname{card} I+$ 1) $* k$ by auto
then show ?case using insert by auto
qed
define $f$ where $f i x=($ case ix of $(i, x) \Rightarrow x)$ for $i x::{ }^{\prime} a \times{ }^{\prime} b$
have preimages-le- $k$ : $\forall x \in f^{\prime}$ ?pairs. card $((f-‘\{x\}) \cap$ ?pairs $) \leq k$ proof
fix $x 0$ assume $x 0$-def: $x 0 \in f^{\text {' }}$ ? pairs
have $(f-‘\{x 0\}) \cap$ ?pairs $=\{(i, x) . i \in I \wedge x \in A i \wedge x=x 0\}$ using $f$-def by auto
moreover have card $\{(i, x) . i \in I \wedge x \in A i \wedge x=x 0\}=\operatorname{card}\{i \in I . x 0 \in A i\}$ using 〈finite $I$ 〉
proof -
have inj-on ( $\lambda i .(i, x 0))\{i \in I . x 0 \in A$ i\} by (meson Pair-inject inj-onI)
moreover have ( $\lambda i$. $(i, x 0)$ )' $\{i \in I . x 0 \in A i\}=\{(i, x) . i \in I \wedge x \in A i \wedge x=x 0\}$
by (rule subset-antisym) blast+
ultimately show ?thesis using card-image by fastforce
qed
ultimately have $1: \operatorname{card}((f-‘\{x 0\}) \cap$ ?pairs $)=\operatorname{card} \quad\{i \in I . x 0 \in A i\}$ by auto
have $\exists i 0 . x 0 \in A$ i0 $\wedge i 0 \in I$ using $x 0$-def $f$-def by auto
then have card $\{i \in I . x 0 \in A i\} \leq k$ using assms by auto
with 1 show card $((f-‘\{x 0\}) \cap$ ?pairs $) \leq k$ by auto
qed
have card ?pairs $\leq \operatorname{card}(f$ ' ?pairs $) * k$
proof -
have finite $\{(i, x) . i \in I \wedge x \in A i\}$ using assms card-pairs not-finite-existsD by fastforce
then show ?thesis using limited-preimages[of $f$ ?pairs $k$, OF preimages-le-k] by auto
qed
then have card $I \leq$ card ( $f$ '?pairs) using card-pairs assms by auto moreover have $f^{\prime}$ ? pairs $=(\bigcup i \in I$. (A i)) using $f$-def [abs-def] by auto
ultimately show ?thesis using $f$-def by auto
qed
In a $m \times n$ latin rectangle each number appears in $m$ columns:
lemma latin-rect-card-col:
assumes latin-rect s m $n x<n$
shows card $\{e$ Col| $e$. $e \in s \wedge e \operatorname{Num}=x\}=m$

```
proof -
    have card {e\ins. e 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\ins.e Num = x \longleftrightarrow fst (num-row e) = x by simp
    have bij-betw num-row {e\ins.e 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 card {e\ins. e Num=x}= card {e Col |e.e\ins\wedgee 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\ins. e Num =x} by (metis (mono-tags, lifting)
inj-onD inj-onI mem-Collect-eq)
    then have inj-on (\lambdae. e Col) {e\ins. e Num = x} unfolding inj-on-def using
without-def by auto
    moreover have (\lambdae.e Col)'{e\ins.e Num = x} = {e Col |e.e\ins^e 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 card $\{e$ Num|e. $e \in s \wedge e \mathrm{Col}=x\}=m$
proof -
have card $\{e \in s . e$ 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: $\forall e \in s . e$ Col $=x \longleftrightarrow$ snd (row-col $e)=x$ by simp
have bij-betw row-col $\{e \in s . e$ Col $=x\} \quad(\{0 . .<m\} \times\{x\})$
using bij-restrict[OF 1 2] cartesian-product-margin2[of $x\{0 . .<n\} \quad\{0 . .<m\}]$ assms by auto
then show ?thesis using card-cartesian-product by (simp add: bij-betw-same-card)
qed
moreover have card $\{e \in s . e \operatorname{Col}=x\}=\operatorname{card}\{e \operatorname{Num} \mid e . e \in s \wedge e \operatorname{Col}=x\}$ 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 Col $=x\}$ by (metis (mono-tags, lifting) inj-onD inj-onI mem-Collect-eq)
then have inj-on ( $\lambda e$. e Num) $\{e \in s . e C o l=x\}$ unfolding inj-on-def using without-def by auto

```
    moreover have \((\lambda e . e N u m) '\{e \in s . e \mathrm{Col}=x\}=\{e N u m \mid e . e \in s \wedge e \mathrm{Col}\)
```

$=x\}$ 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 $\exists s^{\prime} . s \subseteq s^{\prime} \wedge$ latin-square $s^{\prime} n$
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? ?ase 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 $=\lambda j .\{0 . .<n\}-\{e N u m \mid e . e \in s \wedge e C o l=j\}$
- Proof of the hall condition:
have $\forall J \subseteq\{0 . .<n\}$. card $J \leq$ card $(\bigcup j \in J$. ?not-in-column $j)$
proof (rule alli; rule impI)
fix $J$ assume $J$-def: $J \subseteq\{0 . .<n\}$ have $\forall j \in J$. card (?not-in-column $j$ ) $=$ Suc $m$ proof
fix $j$ assume $j$-def: $j \in J$
have $\{e$ Num $\mid e . e \in s \wedge e C o l=j\} \subseteq\{0 . .<n\}$ using atLeastLessThan-iff Suc latin-rect-def partial-latin-square-def by auto
moreover then have finite $\{e \operatorname{Num} \mid e . e \in s \wedge e C o l=j\}$ using finite-subset by auto
ultimately have card (?not-in-column j) $=\operatorname{card}\{0 . .<n\}-\operatorname{card}\{e$ Num $\mid e . e \in s \wedge e C o l=j\}$ using card-Diff-subset[of $\{e \operatorname{Num} \mid e . e \in s \wedge e C o l=j\}$ $\{0 . .<n\}]$ by auto
then show card(?not-in-column j) = Suc m using latin-rect-card-num J-def $j$-def Suc by auto
qed
moreover have $\forall j 0 \in J . \forall x \in$ ?not-in-column $j 0$. card $\{j \in J . x \in$ ?not-in-column $j\} \leq$ Suc $m$ proof (rule ballI; rule ballI)
fix $j 0 x$ assume $j 0 \in J x \in$ ?not-in-column j0
then have card $(\{0 . .<n\}-\{e C o l \mid e . e \in s \wedge e N u m=x\})=$ Suc $m$ proof -
have card $\{e \operatorname{Col} \mid e . e \in s \wedge e N u m=x\}=n-S u c m$ using latin-rect-card-col $\langle x \in$ ?not-in-column j0〉Suc by auto
moreover have $\{e \operatorname{Col} \mid e . e \in s \wedge e N u m=x\} \subseteq\{0 . .<n\}$ using Suc latin-rect-def partial-latin-square-def by auto
moreover then have finite $\{e \operatorname{Col} \mid e . e \in s \wedge e N u m=x\}$ using finite-subset by auto
ultimately show ?thesis using card-Diff-subset[of \{e Col|e. és $\wedge e$ Num $=x\}\{0 . .<n\}]$ using Suc.prems by auto
qed
moreover have $\{j \in J . x \in$ ?not-in-column $j\} \subseteq\{0 . .<n\}-\{e$ Col $\mid e . e \in s$ $\wedge e N u m=x\}$ using Diff-mono J-def using $\langle x \in$ ?not-in-column $j 0\rangle$ by blast ultimately show card $\{j \in J . x \in$ ? not-in-column $j\} \leq$ 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 card $J \leq$ card $(\bigcup j \in J$. ?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: $\forall j \in\{0 . .<n\} . R j \in$ ?not-in-column $j \wedge$ inj-on $R$ $\{0 . .<n\}$ using marriage-HV[of $\{0 . .<n\}$ ?not-in-column] by blast
define new-row where new-row $=(\lambda j$. rec-latin-type $(n-S u c m) j(R j))$ ' $\{0 . .<n\}$
define $s^{\prime}$ where $s^{\prime}=s \cup$ new-row
- $s$ ' is now a latin rect with one more row:
have latin-rect $s^{\prime}(n-m) n$
proof -
- We prove all four criteria specified in the lemma latinrectiff:
have $n-m \leq n$ by auto
moreover have partial-latin-square $s^{\prime} n$
proof -
have inj-on (without Col) $s^{\prime}$ unfolding inj-on-def
proof (rule balli; rule ballI; rule impI)
fix e1 e2 assume $e 1 \in s^{\prime} e 2 \in s^{\prime}$ num-row e1 = num-row e2
then have e1 Num = e2 Num e1 Row = e2 Row using without-def by auto moreover have e1 Col $=$ e2 Col
proof (cases)
assume e1 Row $=n-$ Suc $m$
then have e2 Row $=n-$ Suc $m$ using without-def $\langle n u m$-row e1 $=$ num-row e2 ${ }^{2}$ by auto
have $\forall e \in s$. e Row $<n-$ Suc $m$ using Suc latin-rect-iff by blast
then have e1 $\in$ new-row $e 2 \in$ new-row using $s^{\prime}$-def $\left\langle e 1 \in s^{\prime}\right\rangle\left\langle e 2 \in s^{\prime}\right\rangle$ «e1 Row $=n-$ Suc $m\rangle\langle e 2$ Row $=n-S u c m\rangle$ by auto
then have e1 Num $=R(e 1 \mathrm{Col})$ e2 $\mathrm{Num}=R(e 2 \mathrm{Col})$ using new-row-def by auto
then have $R(e 1 \mathrm{Col})=R(e 2 \mathrm{Col})$ using $\langle e 1 \mathrm{Num}=e 2 \mathrm{Num}$ by auto
moreover have e1 Col <n e2 Col $<n$ using $\langle e 1 \in$ new-row $\langle e 2 \in$

```
new-row> new-row-def by auto
            ultimately show e1 Col = e2 Col using R-def inj-on-def by (metis
(mono-tags, lifting) atLeast0LessThan lessThan-iff)
    next
        assume e1 Row }\not=n-Suc 
        then have e1\ins e2\ins using new-row-def s'-def {e1\ins'\rangle\langlee2\ins'\rangle\langlee1 Row
= e2 Row> by auto
            then show e1 Col=e2 Col using Suc latin-rect-def bij-betw-def by (metis
<num-row e1 = num-row e2`inj-onD)
            qed
            ultimately show e1=e2 using latin-type.induct[of \lambdat. 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' col-num e1 = col-num e2
            then have e1 Col = e2 Col e1 Num = e2 Num using without-def by auto
            moreover have e1 Row = e2 Row
            proof (cases)
            assume e1 Row = n-Suc m
            have }\foralle\ins.e Row < n - Suc m using Suc latin-rect-iff by blas
            then have e2 Num \in ?not-in-column (e2 Col) using R-def new-row-def
                <e1 Col = e2 Col` <e1 Num = e2 Num> using s'-def <e1 \in s'\rangle\langlee1 Row = n -
Suc m> by auto
            then show e1 Row = e2 Row using new-row-def <e1 Row = n - Suc m`
s'-def \langlee2 \in s'\rangle by auto
            next
            assume e1 Row }\not=n-Suc 
            then have e1\ins using new-row-def s'-def \langlee1\ins'\rangle by auto
            then have e2 Num & ?not-in-column (e2 Col) using <e1 Col = e2 Col`
<e1 Num = e2 Num〉 by auto
            then have e2\ins using new-row-def s'-def \langlee2\ins'\rangle R-def by auto
            moreover have inj-on col-num s using Suc.prems latin-rect-def[of s ( }
                - Suc m) n] partial-latin-square-def[of s n] by blast
                    ultimately show e1 Row = e2 Row using Suc latin-rect-def by (metis
<col-num e1 = col-num e2\rangle\langlee1 \in s> inj-onD)
            qed
            ultimately show e1=e2 using latin-type.induct[of \lambdat. 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'row-col e1 = row-col e2
        then have e1 Row = e2 Row e1 Col = e2 Col using without-def by auto
        moreover have e1 Num = e2 Num
        proof (cases)
            assume e1 Row = n-Suc m
            then have e2 Row = n - Suc m using without-def<row-col e1 = row-col
e2> by auto
```

have $\forall e \in s$ ．e Row $<n-$ Suc $m$ using Suc latin－rect－iff by blast
then show $e 1 \mathrm{Num}=e^{2}$ Num using $\left\langle e 1 \mathrm{Col}=e^{2}\right.$ Col〉 using new－row－def $s^{\prime}$－def $\left\langle e 1 \in s^{\prime}\right\rangle\left\langle e 2 \in s^{\prime}\right\rangle\langle e 1$ Row $=n-S u c m\rangle\langle e 2$ Row $=n-S u c m\rangle$ by auto next
assume e1 Row $\neq n-$ Suc $m$
then have $e 1 \in s e 2 \in s$ using new－row－def $s^{\prime}$－def $\left\langle e 1 \in s^{\prime}\right\rangle\left\langle e 2 \in s^{\prime}\right\rangle\langle e 1$ Row $=e 2$ Row $>$ by auto
then show e1 Num＝e2 Num using Suc latin－rect－def bij－betw－def by （metis 〈row－col e1 $=$ row－col e2〉inj－onD）
qed
ultimately show $e 1=e 2$ using latin－type．induct $[o f ~ \lambda t$ ．e1 $t=e 2 t]$ by auto
qed
moreover have $\forall e \in s^{\prime} . \forall t$ ．e $t<n$
proof（rule ballI；rule allI）
fix $e t$ assume $e \in s^{\prime}$
then show $e t<n$
proof（cases）
assume $e \in$ new－row
then show ？thesis using new－row－def $R$－def by（induction t）auto next
assume $e \notin$ new－row
then show ？thesis using $s^{\prime}$－def $\left\langle e \in s^{\prime}\right\rangle$ latin－rect－def partial－latin－square－def
Suc by auto
qed
qed
ultimately show partial－latin－square $s^{\prime} n$ unfolding partial－latin－square－def
using latin－type．induct［of $\lambda$ t．inj－on（without $t$ ）$s$ ］by auto
qed
moreover have card $s^{\prime}=n *(n-m)$
proof－
have card－s：card $s=n *(n-S u c m)$ using latin－rect－iff Suc by auto
have card－new－row：card new－row $=n$ unfolding new－row－def
proof－
have inj－on $(\lambda j$ ．rec－latin－type $(n-S u c m) j(R j))\{0 . .<n\}$ unfolding inj－on－def
proof（rule ballI；rule ballI；rule impI）
fix $j 1 j 2$ assume $j 1 \in\{0 . .<n\} j 2 \in\{0 . .<n\}$ rec－latin－type $(n-S u c m)$
$j 1(R j 1)=$ rec－latin－type $(n-S u c m) j 2(R j 2)$
then show $j 1=j 2$ using latin－type．rec（2）［of $(n-S u c m) j 1 R j 1]$ latin－type．rec（2）［of－j2－］by auto
qed
then show card $((\lambda j$ ．rec－latin－type $(n-S u c m) j(R j)) \cdot\{0 . .<n\})=n$
by（simp add：card－image）
qed
have $s \cap$ new－row $=\{ \}$
proof－
have $\forall e \in s$ ．e Row $<n-$ Suc $m$ using Suc latin－rect－iff by blast
then have $\forall e \in$ 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 card s'= card s+ card 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 }\foralle\in\mp@subsup{s}{}{\prime}.\mathrm{ . e Row < (n-m)
    proof (rule ballI; cases)
            fix e
            assume e\innew-row
            then show e Row<n-m using Suc new-row-def R-def by auto
    next
            fix }
            assume e\in s' e\not\innew-row
            then have e Row < n - Suc m using latin-rect-iff Suc s'-def \langlee\ins'\rangle by
auto
            then show e 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 \(s^{\prime \prime}\) where \(s^{\prime} \subseteq s^{\prime \prime}\) latin-square \(s^{\prime \prime} n\) using Suc by auto
then have \(s \subseteq s^{\prime \prime}\) using \(s^{\prime}\)-def by auto
then show \(\exists s^{\prime} . s \subseteq s^{\prime} \wedge\) latin-square \(s^{\prime} n\) using 〈latin-square \(s^{\prime \prime} n\) 〉 by auto qed
end
```


## References

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

