Inzidenzmatrizen endlicher projektiver Ebenen

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius–Maximilians–Universität Würzburg

vorgelegt von

Helmut Kramer

aus

Mainbernheim

Würzburg 2004

Der gute Christ soll sich hüten vor den Mathematikern und all denen, die leere Vorhersagungen zu machen pflegen, schon gar dann, wenn diese Vorhersagungen zutreffen. Es besteht nämlich die Gefahr, daß die Mathematiker mit dem Teufel im Bund den Geist trüben und den Menschen in die Bande der Hölle verstricken.

Augustinus

Eingereicht am: 22. Oktober 2004

bei der Fakultät für Mathematik und Informatik

1. Gutachter: Prof. Dr. O. Mutzbauer
2. Gutachter: Prof. Dr. H. Heineken

Tag der mündlichen Prüfung: 20. Dezember 2004

Inhaltsverzeichnis

Wer die Geometrie begreift,
vermag in dieser Welt alles zu verstehen.

Galileo Galilei

1 Einleitung

Die projektiven Ebenen sind das wahrscheinlich am meisten untersuchte Objekt der Geometrie. Dennoch ist bis heute keine vollständige Klassifizierung der projektiven Ebenen gelungen. Selbst für endliche Ebenen ist nicht geklärt, zu welchen Ordnungen es projektive Ebenen gibt und zu welchen nicht. Hierzu gibt es lediglich zwei theoretische Ergebnisse, einerseits von Bruck und Ryser, wonach bestimmte Ordnungen nicht auftreten können, und andererseits gibt es zu jeder Primzahlpotenzordnung die desarguessche Ebene, wobei weitere, nicht dazu isomorphe Ebenen dieser Ordnung nicht ausgeschlossen sind.

Alle Ebenen zu einer gegebenen Ordnung können wir erhalten, indem wir einfach alle Möglichkeiten ausprobieren, die Geraden so durch die Punkte zu legen, daß die Axiome der projektiven Ebene erfüllt werden. Dies macht die Frage nach den projektiven Ebenen einer bestimmten Ordnung zu einem Problem der Kombinatorik. Eine Durchführung per Hand führt jedoch schon bald an natürliche Grenzen, so daß eine Berechnung per Computer angemessen erscheint. In diesem Zusammenhang ist durch die Inzidenzmatrix eine geeignete Beschreibung der Geometrie gegeben. Zur Berechnung der projektiven Ebenen höherer Ordnung ist es zudem noch ratsam, die Inzidenzmatrix vorzustrukturieren, in unserem Fall beispielsweise durch Doppelordnung.

Die wichtigsten Ergebnisse in diesem Gebiet sind bisher von Lam [15], welcher per Computer alle Isomorphietypen von projektiven Ebenen der Ordnung 9 berechnete. Hierzu verwendete auch er eine Vorstrukturierung der Inzidenzmatrix, allerdings nicht mit Hilfe der Doppelordnung. Außerdem konnte Lam [16] per Computer zeigen, daß es keine projektive Ebene der Ordnung 10 gibt, allerdings unter starker Ausnutzung der Codierungstheorie und unter Verwendung spezieller Eigenschaften bei dieser Ordnung. Höhere Ordnungen konnten bislang noch nicht per Computer in Angriff genommen werden. Doch auch bei den geklärten Fällen wäre es wünschenswert, eine unabhängige Bestätigung dieser Ergebnisse zu haben, birgt doch jeder Computerbeweis die Gefahr eines falschen Ergebnisses wegen Hardware- und Softwarefehlern, die aufgrund der großen Komplexität kaum zu enttarnen sind.

In unserem Ansatz gehen wir von einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene aus, die durch die Doppelordnung eine besondere Struktur aufweist. Diese Struktur wurde zuerst von M. Mann [19] für eine computerunterstützte Berechnung der projektiven Ebenen genutzt, wobei er jedoch nur die Ebenen bis Ordnung 8 berechnen konnte. Hierauf aufbauend

wollen wir daher den Algorithmus effektiver gestalten, so daß auch Ebenen höherer Ordnung zugängig sind. Dazu untersuchen wir in den ersten Kapiteln die Struktur der doppelgeordneten Inzidenzmatrix noch näher, um sie zu einer weiteren Beschleunigung des Algorithmus nutzbar zu machen. Nebenbei zeigt sich eine bereits von Bose [4] entdeckte Verbindung zu einem anderen Objekt der Kombinatorik, den lateinischen Quadraten. Eine zusätzliche Beschleunigung des Algorithmus verspricht eine verbesserte Datenstruktur.

In Kapitel 5 widmen wir uns dann dem Problem, zu entscheiden, ob zwei Inzidenzmatrizen zu isomorphen Ebenen gehören oder zu nicht-isomorphen. Hierzu gab es ebenfalls bereits ein Programm von B. Schwarz [23], welches jedoch für unsere Bedürfnisse nicht schnell genug war. Abhilfe verschafft ein neuer Algorithmus unter intensiver Nutzung der Doppelordnung. In Kapitel 7 werden wir uns daraufhin speziell mit den desarguesschen Ebenen beschäftigen, zeigen ihre doppelgeordneten Inzidenzmatrizen doch eine besonders schöne Struktur. In Kapitel 8 schließlich werden wir uns noch mit den Blockplänen beschäftigen, welche in gewisser Weise eine Verallgemeinerung der endlichen projektiven Ebenen darstellen.

Ein wesentlicher Teil dieser Arbeit besteht natürlich in der programmiertechnischen Umsetzung der Algorithmen. Die Programme sind alle in C++ geschrieben. Als Compiler nutzen wir den ,,GNU project C++“(g++)-Compiler in der Version 3.3.1-29 unter Linux. Als Referenz bezüglich der Programmierung in C++ sei hierbei auf das Buch von Schildt [21] verwiesen.

An dieser Stelle möchte ich mich noch bei Prof. Dr. O. Mutzbauer herzlich für den Vorschlag dieses schönen, interessanten Themas und seine stets freundliche und hilfreiche Betreuung bedanken.

2 Grundlagen

Dieses Kapitel soll die Grundlagen über projektive Ebenen und die Inzidenzmatrizen bereitstellen.

2.1 Inzidenzstrukturen und projektive Ebenen

Zunächst definieren wir die Grundbegriffe, die Inzidenzstruktur und die projektive Ebene (vgl. Stevenson [24]).

Definition 2.1
Eine **Inzidenzstruktur** ist ein Tripel $(\mathcal{P}, \mathcal{L}, I)$ mit Mengen $\mathcal{P}$ und $\mathcal{L}$ und einer Relation $I \subseteq \mathcal{P} \times \mathcal{L}$. Hierbei heißt I **Inzidenzrelation**.
Die Elemente von $\mathcal{P}$ heißen **Punkte**, die Elemente von $\mathcal{L}$ **Geraden**. Für einen Punkt $p \in \mathcal{P}$ und eine Gerade $l \in \mathcal{L}$ mit pIl sagt man p liegt auf l. Die Menge $X \subseteq \mathcal{P}$ heißt **kollinear**, falls es eine Gerade $l \in \mathcal{L}$ gibt, auf der alle $x \in X$ liegen. Die Menge $Y \subseteq \mathcal{L}$ heißt **konfluent**, falls es einen Punkt $p \in \mathcal{P}$ gibt, der auf allen $y \in Y$ liegt.
Für $p \in \mathcal{P}$ sei weiterhin $\mathcal{L}_p := \{l \in \mathcal{L} : pIl\}$ das Büschel durch p und für $l \in \mathcal{L}$ sei $[l] := \{p \in \mathcal{P} : pIl\}$ die Punktreihe auf l.
Ein Paar (α, β) mit $\alpha \in Sym(\mathcal{P})$ und $\beta \in Sym(\mathcal{L})$ heißt **Kollineation**, wenn gilt $\alpha(p)I\beta(l) \Leftrightarrow pIl$ für alle $p \in \mathcal{P}$ und $l \in \mathcal{L}$. Die Kollineationen einer Inzidenzstruktur bilden eine Gruppe, die **Kollineationsgruppe**.
Vertauschen wir bei einer Inzidenzstruktur $(\mathcal{P}, \mathcal{L}, I)$ Punkte und Geraden, so erhalten wir eine Inzidenzstruktur $(\mathcal{L}, \mathcal{P}, I')$ mit $lI'p :\Leftrightarrow pIl$ für alle $p \in \mathcal{P},\ l \in \mathcal{L}$, die zur ursprünglichen **duale** Inzidenzstruktur.
Eine Inzidenzstruktur heißt endlich, wenn $\mathcal{P}$ und $\mathcal{L}$ endlich sind.

Definition 2.2
Eine projektive Ebene ist eine Inzidenzstruktur mit:

i) Je zwei Punkte liegen auf genau einer Geraden.

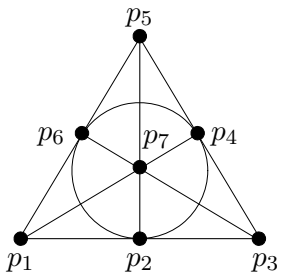
ii) Je zwei Geraden schneiden sich in genau einem Punkt.

iii) Es gibt ein Viereck, d. h. vier Punkte, von denen keine drei kollinear sind.

Man beachte i) ist dual zu ii). Daher ist die zu einer projektiven Ebene duale Inzidenzstruktur ebenfalls eine projektive Ebene. Zu zwei Punkten p_1 und p_2 gibt es nach i) genau eine Gerade l, auf der p_1 und p_2 liegen. Dann nennen wir l die **Verbindungsgerade** von p_1 und p_2 und schreiben $p_1 \vee p_2$ für l. Analog gibt es nach ii) zu zwei Geraden l_1 und l_2 genau einen Punkt p, der auf l_1 und l_2 liegt. Dann heißt p **Schnittpunkt** von l_1 und l_2 und wir schreiben $l_1 \wedge l_2$ für p. Für endliche Ebenen gelten weitere Besonderheiten (vgl. Stevenson [24]).

Satz 2.3
Sei $(\mathcal{P}, \mathcal{L}, I)$ eine endliche projektive Ebene. Dann gibt es ein $n \in \mathbb{N}$, $n \geq 2$ mit $|\mathcal{L}_p| = n+1 = |[l]|$ für alle $l \in \mathcal{L},\ p \in \mathcal{P}$. Weiterhin ist $|\mathcal{P}| = n^2+n+1 = |\mathcal{L}|$. Man nennt dann n die **Ordnung** *der projektiven Ebene.*

Beispiel 2.4
Die Existenz projektiver Ebenen zeigt das folgende Beispiel:



$$l_1 = \{p_1, p_2, p_3\}$$
$$l_2 = \{p_5, p_6, p_1\}$$
$$l_3 = \{p_3, p_4, p_5\}$$
$$l_4 = \{p_1, p_7, p_4\}$$
$$l_5 = \{p_3, p_7, p_6\}$$
$$l_6 = \{p_5, p_7, p_2\}$$
$$l_7 = \{p_2, p_4, p_6\}$$

Es läßt sich leicht nachprüfen, daß diese Geometrie alle Axiome der projektiven Ebene erfüllt, also eine projektive Ebene ist. Auf jeder Geraden liegen 3 Punkte, daher ist die Ordnung 2.
Weitere Ebenen bekommen wir durch folgendes Verfahren:
Sei $\mathbb{K}$ ein Körper, $\mathcal{P}$ die Menge der eindimensionalen Unterräume von $\mathbb{K}^3$ und $\mathcal{L}$ die Menge der zweidimensionalen Unterräume von $\mathbb{K}^3$. Dann ist $P(\mathbb{K}) := (\mathcal{P}, \mathcal{L}, \subseteq)$ eine projektive Ebene, die **desarguessche projektive Ebene** über dem Körper $\mathbb{K}$. Ist die Ebene endlich, also $\mathbb{K}$ endlich, so erhält man als Ordnung der Ebene gerade die Ordnung des Körpers $\mathbb{K}$. Unsere obige projektive Ebene ist beispielsweise die Ebene über dem Körper $\mathbb{F}_2$.

Aus der Körpertheorie ist bekannt, daß es zu jeder Primzahlpotenz q einen Körper der Ordnung q gibt, nämlich das Galoisfeld $\mathbb{F}_q$. Somit existiert zu jeder Primzahlpotenz q auch eine projektive Ebene der Ordnung q. Demnach gibt es zu den Ordnungen $n = 2, 3, 4, 5, 7, 8, 9, 11, 13$ projektive Ebenen. Nun stellt sich die Frage, ob es zu allen $n \in \mathbb{N}$ eine projektive Ebene der Ordnung n gibt. Eine teilweise Antwort gibt folgender Satz (vgl. Bruck, Ryser [6]).

Satz 2.5 (Bruck-Ryser)
Sei $n \equiv 1, 2 \pmod 4$ und der quadratfreie Teil von n habe mindestens einen Primfaktor p mit $p = 3 \pmod 4$. Dann existiert keine projektive Ebene der Ordnung n.

Eine häufig verwandte, äquivalente Formulierung ist:

Satz 2.6
Sei $n \equiv 1, 2 \pmod 4$. Gibt es eine projektive Ebene der Ordnung n, so gibt es zwei Zahlen $x, y \in \mathbb{N}$ mit $n = x^2 + y^2$.

Nach diesem Satz gibt es somit keine projektive Ebene der Ordnung 6. Der kleinste noch ungeklärte Fall ist demnach Ordnung 10, denn es ist zwar $10 = 2 \mod 4$, doch es ist auch $10 = 1^2 + 3^2$, weshalb der Satz von Bruck-Ryser nicht anwendbar ist. Erst W. C. H. Lam [16], einem Schüler von Ryser, ist es 1989 in einem aufsehenerregenden Computerbeweis gelungen, zu zeigen, daß es keine projektive Ebene der Ordnung 10 gibt. Dieser Computerbeweis war sehr rechenintensiv und benötigte ca. 2000 Stunden Rechenzeit eines CRAY-Rechners. Ist n eine Primzahlpotenz, so ist die Existenz einer projektiven Ebene geklärt, dennoch stellt sich die Frage, ob es nicht vielleicht mehrere Ebenen gibt. Bis zur Ordnung 8 gibt es bis auf Isomorphie jeweils nur eine projektive Ebene. Bei Ordnung 9 sind 4 verschiedene projektive Ebenen bekannt: Die desarguessche Ebene, die Links-Fastkörper-Ebene, die Rechts-Fastkörper-Ebene und die Hughes-Ebene. Durch einen rechenintensiven Computerbeweis zeigten Lam, Kolesova und Thiel [15], daß dies auch schon alle sind.

2.2 Inzidenzmatrizen

Ist eine Inzidenzstruktur endlich, so läßt sich die Inzidenzrelation in eine Matrix A auf folgende Weise umformen: Die Zeilen der Matrix werden als Geraden aufgefaßt und die Spalten der Matrix als Punkte. Liegt nun ein Punkt auf einer Geraden, so schreibt man an die entsprechende Stelle von A eine 1, ansonsten eine 0. Man nennt dann A die **Inzidenzmatrix** zur Inzidenzstruktur $(\mathcal{P}, \mathcal{L}, I)$. Die **Linien** einer Inzidenzmatrix bezeichnen die Zeilen oder die Spalten der Inzidenzmatrix. Wir sagen zwei Linien **schneiden sich m-fach**, wenn ihr Skalarprodukt $m > 0$ ergibt. Der Einfachheit halber identifizieren wir die Zeilen bzw. Spalten der Inzidenzmatrix mit den zugehörigen Geraden bzw. Punkten der Inzidenzstruktur und übertragen damit auch die Sprache der Geometrie auf die Inzidenzmatrix. Zudem bezeichnen wir in Zukunft standardmäßig die i-te Zeile der Inzidenzmatrix mit l_i und die j-te Spalte mit p_j. Zwei Inzidenzmatrizen A und B heißen **permutationsäquivalent**, wenn sie durch Permutation der Linien ineinander übergeführt werden können, d. h. es gibt Permutationsmatrizen P und Q mit $PAQ^\top = B$. In diesem Fall heißt das Paar (P, Q) die **Permutationsäquivalenz** von A zu B.

Man beachte: Die Permutationsäquivalenz zweier Inzidenzmatrizen bedeutet, daß sie von derselben Inzidenzstruktur stammen, denn Inzidenzmatrizen sind nur bis auf Permutation der Linien eindeutig. Weiterhin übersetzt sich eine Kollineation bei der zugehörigen Inzidenzmatrix A zu einem Paar (P, Q) von Permutationsmatrizen mit $PAQ^\top = A$. Hierbei ist Q durch P eindeutig vorgegeben, da A invertierbar ist (zumindest über $\mathbb{Q}$). Ist A die Inzidenzmatrix zu einer Inzidenzstruktur, so ist $A^\top$ die Inzidenzmatrix zur dualen Inzidenzstruktur. Die Anzahl der verschiedenen Inzidenzmatrizen

läßt sich durch folgendes Lemma mit der Kollineationsgruppe in Verbindung bringen:

Lemma 2.7
Sei I eine Inzidenzmatrix vom Format $n \times m$ und G ihre Kollineationsgruppe. Dann ist der Index der Kollineationsgruppe $[\mathcal{S}_n \times \mathcal{S}_m : G] = \frac{n!\,m!}{|G|}$ gerade die Anzahl der zu I permutationsäquivalenten Inzidenzmatrizen. Ist $I' := P_0 I Q_0^\top$ eine dieser zu I permutationsäquivalenten Inzidenzmatrizen mit der Permutationsäquivalenz (P_0, Q_0), so ist $\{(P,Q) \in \mathcal{S}_n \times \mathcal{S}_m \mid PIQ^\top = I'\} = \{(P_0P', Q_0Q') \mid (P', Q') \in G\}$. Insbesondere ist die Anzahl der Permutationsäquivalenzen von I zu I' gerade die Ordnung der Kollineationsgruppe.

Beweis:

Die Gruppe $\mathcal{S}_n \times \mathcal{S}_m$ wirkt transitiv auf der Menge M der zu I permutationsäquivalenten Inzidenzmatrizen per Abbildung $(P,Q) \mapsto PIQ^\top$. Der Stabilisator $\mathrm{Stab}_I(\mathcal{S}_n \times \mathcal{S}_m)$ dieser Wirkung ist gerade die Kollineationsgruppe G von I und es ist daher $|M| = [\mathcal{S}_n \times \mathcal{S}_m : \mathrm{Stab}_I(\mathcal{S}_n \times \mathcal{S}_m)] = [\mathcal{S}_n \times \mathcal{S}_m : G] = \frac{n!\,m!}{|G|}$ der Stabilisatorindex.

Sei nun $I' = P_0 I Q_0^\top$. Gilt $(P', Q') \in G$, so ist $(P_0P')I(Q_0Q')^\top = P_0(P'IQ'^\top)Q_0^\top = P_0IQ_0^\top = I'$, also (P_0P', Q_0Q') eine Permutationsäquivalenz von I zu I'. Gilt umgekehrt $PIQ^\top = I'$, so ist $(P_0^\top P)I(Q_0^\top Q)^\top = P_0^\top(PIQ^\top)Q_0 = P_0^\top I' Q_0 = I$, also $(P,Q) = (P_0P', Q_0Q')$ mit $(P', Q') \in G$.

□

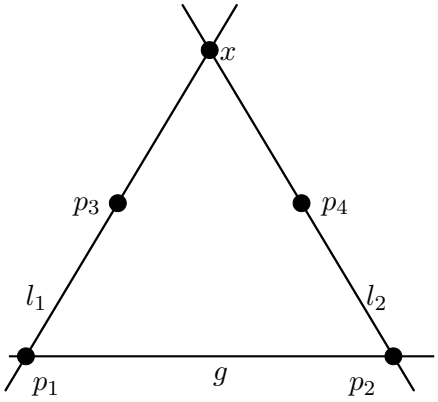
Nach diesem Lemma gibt es also umso mehr Inzidenzmatrizen zu einer gegebenen Inzidenzstruktur, je weniger Kollineationen diese hat. Dies liefert uns ein Maß, wie wahrscheinlich es ist, gewissermaßen „per Zufall“ wenigstens eine dieser Inzidenzmatrizen zu finden. Umgekehrt sichert uns eine große Kollineationsgruppe eine Vielzahl von Permutationsäquivalenzen zweier zugehöriger Inzidenzmatrizen. Daher ist in diesem Fall zu erwarten, daß sich verhältnismäßig schnell eine dieser Permutationsäquivalenzen finden läßt. Kehren wir zurück zu den projektiven Ebenen, so lassen sich ihre Axiome wie folgt in Axiome ihrer Inzidenzmatrix übertragen (vgl. Bruck, Ryser [6]).

Satz 2.8
Eine Matrix $A \in \{0,1\}^{m \times m}$, ist genau dann eine Inzidenzmatrix einer projektiven Ebene, wenn es ein $n \in \mathbb{N}$, $n \geq 2$ gibt mit $AA^\top = A^\top A = nE + J$ mit der Einheitsmatrix E und der Einsmatrix $J = (j_{h,k})_{1 \leq h,k \leq m}$, wobei $j_{h,k} = 1$ für alle $h,k = 1, \ldots, m$. Dann ist $m = n^2 + n + 1$ und n ist die Ordnung der projektiven Ebene.

Beweis:

Ist A eine Inzidenzmatrix einer projektiven Ebene der Ordnung n, so betragen die Zeilensummen jeweils $n+1$. Daher ist das Skalarprodukt einer Zeile mit sich selbst stets $n+1$. Dies ergibt die Diagonalelemente von $AA^\top$. Dagegen ist das Skalarprodukt zweier verschiedener Zeilen immer 1, da zwei verschiedene Geraden sich stets in genau einem Punkt schneiden. Dies ergibt die Nichtdiagonalelemente von $AA^\top$. Die duale Aussage ergibt die Form von $A^\top A$.

Sei nun umgekehrt $AA^\top = A^\top A = nE + J$. Dann ist das Skalarprodukt zweier verschiedener Linien immer 1, d. h. bei der zugrundeliegenden Inzidenzstruktur schneiden sich zwei verschiedene Geraden stets in genau einem Punkt und zwei verschiedene Punkte lassen sich durch genau eine Gerade verbinden, es gelten also i) und ii). Für die Existenz des Vierecks benötigen wir, daß $n \geq 2$. Dann ist nämlich das Skalarprodukt einer Linie mit sich selbst immer mindestens 3, d. h. durch jeden Punkt gehen mindestens drei Geraden und, dual dazu, auf jeder Geraden liegen mindestens drei Punkte. Betrachten wir hierzu eine Gerade g. Auf dieser liegen zwei Punkte p_1 und p_2. Da durch jeden Punkt drei Geraden gehen, finden wir eine Gerade $l_1 \neq g$ durch p_1 und $l_2 \neq g$ durch p_2. Sei x der Schnittpunkt von l_1 und l_2. Da auf jeder Geraden drei Punkte liegen, existieren die Punkte $p_3 \neq p_1, x$ auf l_1 und $p_4 \neq p_2, x$ auf l_2. Dann bilden p_1, p_2, p_3 und p_4 ein Viereck.



□

Beispiel 2.9
Betrachten wir die zu unserem Beispiel 2.4 gehörende Inzidenzmatrix:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
l_1	1	1	1	0	0	0	0
l_2	1	0	0	0	1	1	0
l_3	0	0	1	1	1	0	0
l_4	1	0	0	1	0	0	1
l_5	0	0	1	0	0	1	1
l_6	0	1	0	0	1	0	1
l_7	0	1	0	1	0	1	0

Wir sehen, auf jeder Linie sind drei Einsen und je zwei verschiedene Linien haben als Skalarprodukt 1.

2.3 Doppelordnung

Mit den bisherigen Erkenntnissen können wir bereits einen einfachen Algorithmus zur Berechnung einer Inzidenzmatrix einer projektiven Ebene erstellen:
Wir verteilen in einer leeren $\big((n^2+n+1)\times(n^2+n+1)\big)$-Matrix zeilenweise 0, bis wir an einer Stelle zu einem Widerspruch kommen, sei es, daß die Liniensumme einer schon vollständig besetzten Linie nicht $n+1$ ist, oder das Skalarprodukt zweier verschiedener schon vollständig besetzter Linien nicht 1 ergibt. In diesem Fall löschen wir die zuletzt belegten Stellen, bis wir an eine Stelle gelangen, an der wir widerspruchsfrei den Eintrag von 0 auf 1 ändern können. Diesen Eintrag ändern wir dann und beginnen den Algorithmus ab dieser Stelle von neuem. Dies ist für höhere Ordnungen ein sehr rechenintensiver Algorithmus, zumal hierdurch auch alle zu einer Inzidenzmatrix permutationsäquivalenten Inzidenzmatrizen berechnet werden. Daher ist die Idee naheliegend, die Inzidenzmatrix geeignet vorzustrukturieren, um einerseits den Algorithmus zu beschleunigen, indem schon zu Beginn möglichst viele Stellen festgelegt werden, aber andererseits auch sicherzustellen, daß zu jeder Inzidenzmatrix eine permutationsäquivalente Inzidenzmatrix erzeugt wird. Eine Möglichkeit bildet die Doppelordnung.

Definition 2.10
Seien R eine totalgeordnete Menge und $a, b \in R^n$. Die Zuweisung $a < b :\Leftrightarrow$ für den kleinsten Index i mit $a_i \neq b_i$ gilt $a_i < b_i$ erzeugt eine Totalordnung auf R^n, die **lexikographische Ordnung**.
Eine Matrix mit Einträgen aus R heißt **doppelgeordnet**, falls die Zeilen und Spalten der Matrix absteigend von oben nach unten bzw. links nach rechts lexikographisch geordnet sind.

Die Existenz einer permutationsäquivalenten doppelgeordneten Matrix ist nach A. Mader und O. Mutzbauer [18] für jede $(0,1)$-Matrix gesichert, wo-

durch die Doppelordnung zu einer besonders interessanten Möglichkeit zur Vorstrukturierung wird. Doppelgeordnete Inzidenzmatrizen von projektiven Ebenen weisen folgende Struktur auf:

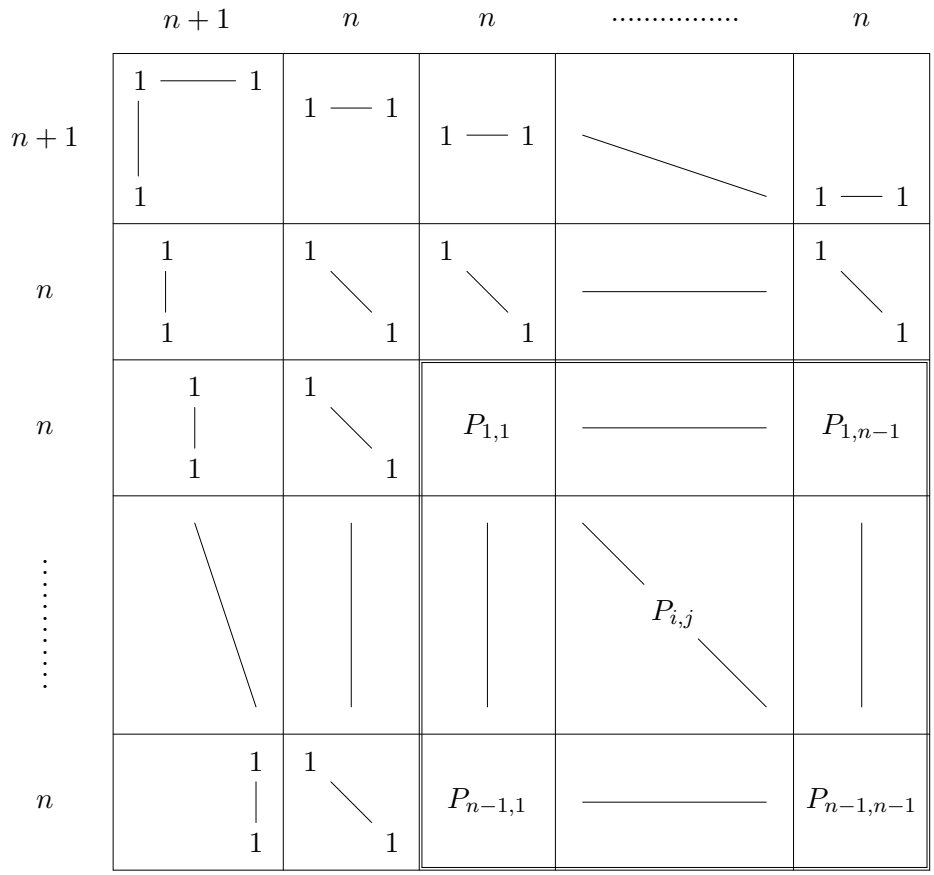


Abbildung 1: doppelgeordnete Inzidenzmatrix

Hierbei sind $P_{i,j}$ für $1 \leq i, j \leq n-1$ jeweils Permutationsmatrizen der Größe n. Um eine strukturiertere Darstellung der doppelgeordneten Inzidenzmatrix zu ermöglichen, treffen wir noch folgende Vereinbarungen: Sei im folgenden die $(n+1) \times (n+1)$-Matrix $\Gamma = (\gamma_{i,j})$ definiert durch $\gamma_{1,j} = 1 = \gamma_{j,1}$ für $1 \leq j \leq n+1$ und $\gamma_{i,j} = 0$ sonst. Weiterhin sei die $n \times (n+1)$-Matrix $\Theta_{i_0} = (\vartheta_{i,j})$ definiert durch $\vartheta_{i_0,j} = 1$ für $1 \leq j \leq n$ und $\vartheta_{i,j} = 0$ sonst. Dann heißt Θ_{i_0} die **i_0-te Zeilenstufen-Matrix**. Analog heißt $\Theta_{i_0}^{\top}$ die **i_0-te Spaltenstufen-Matrix**. Damit hat eine doppelgeordnete Inzidenzmatrix einer projektiven Ebene die Form

$$\begin{bmatrix} \Gamma & \Theta_2 & \cdots & \cdots & \Theta_{n+1} \\ \Theta_2^\top & E & \cdots & \cdots & E \\ \vdots & \vdots & P_{1,1} & \cdots & P_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Theta_{n+1}^\top & E & P_{n-1,1} & \cdots & P_{n-1,n-1} \end{bmatrix}$$

mit den Permutationsmatrizen $(P_{i,j})_{1\leq i,j\leq n-1}$. Hierbei ist ein **Block** eine der Matrizen $P_{i,j}$ mit $1 \leq i,j \leq n-1$, eine der Einheitsmatrizen E, eine der Stufenmatrizen Θ_k oder $\Theta_k^\top$ mit $2 \leq k \leq n+1$ oder aber Γ. Die von Γ und den Zeilenstufen-Matrizen gebildeten Zeilen bezeichnen wir als die **Rand-Blockzeile** und die von Γ und den Spaltenstufen-Matrizen gebildeten Spalten als die **Rand-Blockspalte**. Der **Rand** der Matrix schließlich besteht aus der Rand-Blockspalte und der Rand-Blockzeile. Die durch Entfernen des Randes übrig bleibende Matrix heißt **Kern** der Matrix. Setzen wir $P_{0,i} = E = P_{i,0}$ für $0 \leq i \leq n-1$, so hat der Kern die Form $(P_{i,j})_{0\leq i,j\leq n-1}$. Weiterhin bezeichnen wir $(\Theta_{i_0+2}^\top, (P_{i_0,j})_{0\leq j\leq n-1})$ als die **i_0-te Blockzeile** und $(\Theta_{j_0+2}, (P_{i,j_0})_{0\leq i\leq n-1})$ als die **j_0-te Blockspalte**. Häufig ist die Zeilenstufen-Matrix bzw. Spaltenstufen-Matrix für unsere Betrachtungen nicht wichtig, weshalb wir sie dann auch weglassen. Der Begriff **Blocklinie** bezeichnet außerdem entweder eine Blockspalte oder eine Blockzeile. Die i_0-te Blocklinie nennen wir für $i_0 \geq 1$ auch **innere Blocklinie**. Wenn wir die Blöcke einer Blocklinie mit einer Permutationsmatrix von links oder rechts multiplizieren oder aber konjugieren, sprechen wir der Kürze halber auch davon, daß wir die Blocklinie von links oder von rechts mit der Permutationsmatrix multiplizieren oder aber konjugieren. Eine Zeile eines Blockes heißt **Zeilenabschnitt** und eine Spalte eines Blockes **Spaltenabschnitt**. Die Zeilennummer einer Zeile innerhalb seiner Blockzeile nennen wir **relative Zeilennummer** und analog die Spaltennummer einer Spalte innerhalb seiner Blockspalte **relative Spaltennummer**. Zuletzt bezeichnen wir noch mit **Permutationsteil** die durch $(P_{i,j})_{1\leq i,j\leq n-1}$ gegebene Teilmatrix. Sie entsteht aus dem Kern durch „Weglassen des Randes aus Einheitsmatrizen“.

Bisher haben wir nur einige Begriffe zu besseren Beschreibung der doppelgeordneten Inzidenzmatrix bereitgestellt, jedoch noch nicht sichergestellt, daß die Matrix auch tatsächlich diese Form hat. Um dies beweisen zu können und um die genauen Eigenschaften dieser Inzidenzmatrix beschreiben zu können, benötigen wir noch folgende Definition.

Definition 2.11
Sei $M := (m_{i,j})$ eine $(0,1)$-Matrix. Man sagt, M erfüllt die **Rechtecksregel**, wenn die folgende Konstellation nicht auftritt:

$$\begin{matrix} 1 & \cdots & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & 1 \end{matrix}$$

Das heißt, zu je vier Indizes i, j, k, l, mit $i \neq j$ und $k \neq l$, darf niemals $m_{i,k}m_{i,l}m_{j,k}m_{j,l} = 1$ sein. Die Rechtecksregel besagt gerade, daß sich zwei verschiedene Linien nicht mehrmals schneiden dürfen. Daher gilt sie auch in jeder Inzidenzmatrix einer projektiven Ebene. Weiterhin erfüllt mit M auch jede Teilmatrix von M die Rechtecksregel.
Eine Menge von $(0,1)$-Matrizen von gleichem Format heißt **überdeckungsfrei**, wenn deren Summe über $\mathbb{Z}$ wieder eine $(0,1)$-Matrix ist. Man beachte hierbei, daß die Überdeckungsfreiheit mit der paarweisen Überdeckungsfreiheit übereinstimmt.
Eine Permutationsmatrix heißt **fixpunktfrei**, falls sie mit der Einheitsmatrix eine überdeckungsfreie Menge bildet. Sind $P_1, \ldots, P_m$ überdeckungsfreie $(n \times n)$-Permutationsmatrizen, so ist $m \leq n$. Im Falle $n = m$ heißt $\{P_1, \ldots, P_n\}$ **flächendeckend** bzw. ein **flächendeckendes Ensemble**. Dies ist äquivalent zu

$$\sum_{i=1}^{n} P_i = J.$$

Hierbei ist $J = (j_{l,m})_{1 \leq l,m \leq n}$ die Einsmatrix, definiert durch $j_{l,m} = 1$ für alle $l, m = 1, \ldots, n$. Eine alternative Charakterisierung eines flächendeckenden Ensembles liefert folgendes Lemma:

Lemma 2.12
Sei M eine Menge von Permutationsmatrizen der Größe n. Genau dann bildet M ein flächendeckendes Ensemble, wenn M scharf transitiv auf den Einheitsvektoren $e_1, \ldots, e_n$ wirkt.

Beweis:

Für eine Permutationsmatrix $P = (p_{i,j})$ gilt $p_{i,j} = 1 \Leftrightarrow Pe_j = e_i$. Ist daher $M = \{P_1, \ldots, P_n\}$ ein flächendeckendes Ensemble, so gibt es zu jedem $i, j = 1, \ldots, n$ genau eine Permutationsmatrix $P = (p_{i,j}) \in M$ mit $p_{i,j} = 1$, also $Pe_j = e_i$. Somit wirkt M scharf transitiv auf $e_1, \ldots, e_n$. Wirkt umgekehrt M scharf transitiv auf $e_1, \ldots, e_n$, so gibt es zu jedem $i, j = 1, \ldots, n$ genau eine Permutationsmatrix $P = (p_{i,j}) \in M$ mit $Pe_j = e_i$, also $p_{i,j} = 1$. Somit ist $\sum_{P \in M} P = J$ mit der Einsmatrix J. Dann ist jedoch $|M| = n$ und M ein flächendeckendes Ensemble.

□

Mit diesen Begriffen können wir nun die genauen Eigenschaften einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene beschreiben. Hierzu dient folgendes, schon länger bekanntes Ergebnis (vgl. J. Huber [10]), welches aus Gründen der Vollständigkeit an dieser Stelle nochmals bewiesen wird.

Satz 2.13
Eine Matrix I ist genau dann eine doppelgeordnete Inzidenzmatrix einer projektiven Ebene der Ordnung $n \geq 2$, wenn gilt:

$$I = \begin{bmatrix} \Gamma_n & \Theta_2 & \cdots & \cdots & \Theta_{n+1} \\ \Theta_2^\top & E & \cdots & \cdots & E \\ \vdots & \vdots & P_{1,1} & \cdots & P_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Theta_{n+1}^\top & E & P_{n-1,1} & \cdots & P_{n-1,n-1} \end{bmatrix},$$

der Permutationsteil erfüllt die Rechtecksregel und die $P_{i,j}$ sind fixpunktfreie Permutationsmatrizen, von denen je zwei in derselben Blocklinie überdeckungsfrei sind.

Man beachte, die Bedingung der Fixpunktfreiheit und der Überdeckungsfreiheit je zweier Permutationsmatrizen des Permutationsteils in derselben Blocklinie bedeutet, daß die inneren Blocklinien jeweils ein flächendeckendes Ensemble bilden.

Beweis:
$\Rightarrow$: Sei zunächst I eine Inzidenzmatrix einer projektiven Ebene der Ordnung n. Da jede Linie $n+1$ Einsen enthält und die Rechtecksregel gewahrt werden muß, bekommt der Rand durch die Doppelordnung die dargestellte Form. Die Zeilenabschnitte des Kerns können nicht mehr als eine Eins enthalten, da dies ansonsten der Rechtecksregel bezüglich der Zeilenstufen-Matrix derselben Blockspalte widerspräche. Andererseits muß jeder Zeilenabschnitt eine Eins enthalten, da in jeder Zeile des Kerns n Einsen auf n Zeilenabschnitte zu verteilen sind. Somit enthält jeder Zeilenabschnitt, und dual dazu auch jeder Spaltenabschnitt, des Kerns genau eine Eins. Dies heißt jedoch gerade, daß die Blöcke des Kerns aus Permutationsmatrizen bestehen. Durch die Doppelordnung werden dann die Blöcke der 0-ten Blockspalte und 0-ten Blockzeile zu Einheitsmatrizen normiert. Diese Struktur der 0-ten Blockzeile erzwingt durch die Rechtecksregel weiterhin die Überdeckungsfreiheit der anderen Blockzeilen, insbesondere, da die Blöcke der 0-ten Blockspalte aus Einheitsmatrizen bestehen, die Fixpunktfreiheit der Blöcke des Permutationsteils. Dual dazu gilt diese Aussage auch für die Blöcke des Permutationsteils in einer Blockspalte. Die Gültigkeit der Rechtecksregel auf dem Permutationsteil schließlich folgt direkt aus deren Gültigkeit in I.
$\Leftarrow$: Erfüllt nun umgekehrt I die Bedingungen des Satzes, so müssen wir

nach Satz 2.8 zum Einen zeigen, daß die Liniensummen $n+1$ sind, und zum Anderen, daß sich zwei verschiedene Linien jeweils genau einmal schneiden. Die erste Aussage folgt unmittelbar aus der Struktur der Matrix, weshalb wir uns sogleich mit der zweiten beschäftigen. Zunächst erfüllt ganz I die Rechtecksregel, denn sie gilt nach Voraussetzung zumindest für den Permutationsteil, nehmen wir noch die 0-ten Blocklinien hinzu, so folgt die Rechtecksregel daraus, daß die Blöcke einer inneren Blocklinie überdeckungsfrei sind, nehmen wir dann noch den Rand hinzu, so folgt die Rechtecksregel aus der Eigenschaft, daß die Blöcke der Kerns aus Permutationsmatrizen bestehen. Durch die Gültigkeit der Rechtecksregel auf ganz I ist nun sichergestellt, daß sich zwei verschiedene Linien nie öfter als einmal schneiden, weshalb wir nur noch zeigen müssen, daß sie sich überhaupt schneiden. Ist hierbei eine der beiden Linien eine der Rand-Blocklinien, so ist dies aus der Struktur des Permutationsteils klar, ebenso bei zwei Linien derselben Blocklinie. Seien daher nun l und l' zwei Linien, nicht aus derselben Blocklinie. Hierbei beschränken wir uns auf den Teil der Linie im Kern, da der Rand in diesem Fall nichts zum Skalarprodukt beiträgt. Seien $(l_i)_{1\leq i\leq n}$ die Linien der Blocklinie von $l' = l_r$. Da die Blöcke Permutationsmatrizen sind, ist $\sum_{i=1}^{n} l_i = e$, der Vektor, der nur 1 als Eintrag enthält. Nun enthält l gerade n Einsen im Kern, weshalb $\sum_{i=1}^{n} \langle l, l_i\rangle = \langle l, \sum_{i=1}^{n} l_i\rangle = \langle l, e\rangle = n$. Da aufgrund der Rechtecksregel jeweils $\langle l, l_i\rangle \leq 1$ gilt, muß $\langle l, l'\rangle = 1$ sein. Somit ist I eine Inzidenzmatrix einer projektiven Ebene.

□

Beispiel 2.14
Betrachten wir die Doppelordnung unserer Inzidenzmatrix aus dem Beispiel 2.9.

1	1	1	0	0	0	0
1	0	0	0	1	1	0
0	0	1	1	1	0	0
1	0	0	1	0	0	1
0	0	1	0	0	1	1
0	1	0	0	1	0	1
0	1	0	1	0	1	0

$\xrightarrow[\text{ordnen}]{\text{spaltenweise}}$

1	1	1	0	0	0	0
1	0	0	1	1	0	0
0	1	0	1	0	1	0
1	0	0	0	0	1	1
0	1	0	0	1	0	1
0	0	1	1	0	0	1
0	0	1	0	1	1	0

1	1	1	0	0	0	0
1	0	0	1	1	0	0
0	1	0	1	0	1	0
1	0	0	0	0	1	1
0	1	0	0	1	0	1
0	0	1	1	0	0	1
0	0	1	0	1	1	0

$\xrightarrow[\text{ordnen}]{\text{zeilenweise}}$

1	1	1	0	0	0	0
1	0	0	1	1	0	0
1	0	0	0	0	1	1
0	1	0	1	0	1	0
0	1	0	0	1	0	1
0	0	1	1	0	0	1
0	0	1	0	1	1	0

Der Permutationsteil besteht in diesem Fall aus nur einer Permutationsmatrix, nämlich $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

Bei höheren Ordnungen läßt sich der Permutationsteil noch weiter festlegen. Vertauschen wir nämlich die i-te und die j-te Blockspalte, $1 \leq i, j \leq n-1$, so läßt sich dies durch Vertauschen der $(i+2)$-ten und $(j+2)$-ten Zeile ausgleichen, so daß die Matrix wieder doppelgeordnet ist. Somit läßt sich das Vertauschen zweier Blocklinien des Permutationsteils permutationsäquivalent gestalten. Daher können wir den Permutationsteil wie folgt festlegen:

$0\ 1\ 0\ \cdots\ 0$	$0\ 0\ 1\ \cdots\ 0$	$\cdots$	$0\ 0\ 0\ \cdots\ 1$
$0\ 0\ 1\ \cdots\ 0$			
$\vdots$			
$0\ 0\ 0\ \cdots\ 1$			

Ist durch die Permutationsmatrizenstruktur, die Fixpunktfreiheit, die Überdeckungsfreiheit oder die Rechtecksregel, an einer Stelle eine Eins verboten, also eine Null erzwungen, so sprechen wir von einer **erzwungenen Null**. Sind in einem Zeilenabschnitt oder einem Spaltenabschnitt $n-1$ Nullen, so muß an der noch freien Stelle eine Eins stehen. Diese nennt man dann **erzwungene Eins**. Mit diesen Termini läßt sich folgender Algorithmus zur Berechnung eines Permutationsteils formulieren:

Algorithmus 2.15

i) Beginnend von links oben, suche zeilenweise die nächste freie Stelle und besetze sie mit 1.

ii) Setze die erzwungenen Nullen.

iii) Tritt dabei ein Widerspruch auf, daß eine erzwungene Null auf eine schon mit Eins besetzte Stelle trifft, oder daß ein Zeilenabschnitt oder Spaltenabschnitt nur aus Nullen besteht, gehe eine Stufe zurück und überschreibe die letzte gesetzte Eins durch Null. Suche die nächste freie Stelle in diesem Zeilenabschnitt, besetze sie mit Eins und fahre mit ii) fort. Gibt es keine freie Stelle mehr in diesem Zeilenabschnitt, so wiederhole diesen Schritt. Ist dies nicht mehr möglich, da man schon am Anfang ist, so ist der Algorithmus ohne Ergebnis zu Ende.

iv) Setze die erzwungenen Einsen. Falls welche gefunden werden, gehe zurück zu Schritt ii) und setze dort die durch die neuen Einsen erzwungenen Nullen.

v) Suche zeilenweise die nächste unbesetzte Stelle, besetze diese mit 1 und gehe zu ii). Sind alle Stellen besetzt, so liefert dieser Algorithmus eine Inzidenzmatrix einer projektiven Ebene.

Dieser Algorithmus arbeitet zeilenweise. Dies ist nicht zwingend, man könnte völlig analog auch spaltenweise vorgehen, es wurde jedoch das zeilenweise Vorgehen, der Präferenz in der Literatur folgend, gewählt. Ein Computerprogramm mit diesem Algorithmus wurde von M. Mann [19] verwirklicht, allerdings stellte es sich als recht langsam heraus, so daß nur Inzidenzmatrizen bis Ordnung 7 in akzeptabler Zeit berechnet werden konnten. Abhilfe verspricht eine verbesserte Datenstruktur. Aufgrund der Permutationsmatrizenstruktur ist nämlich jeder Zeilenabschnitt ein Einheitsvektor $e_i^\top$. Anstatt nun den ganzen Vektor abzuspeichern, können wir ebenso gut lediglich i abspeichern. Damit haben wir für den Permutationsteil anstatt einer $(n(n-1) \times n(n-1))$-Matrix nur noch eine $(n(n-1) \times (n-1))$-Matrix. Dies bedingt natürlich auch eine Modifizierung des Algorithmus:

Algorithmus 2.16

i) Beginne links oben.

ii) Setze an der aktuellen Stelle eine 1.

iii) Kommt es zu einem Widerspruch mit der Fixpunktfreiheit, der Überdeckungsfreiheit oder der Rechtecksregel, so erhöhe den aktuellen Wert um 1. Ist der Wert dann nicht größer als n, gehe zur nächsten Stelle und kehre zu ii) zurück. Ansonsten gehe eine Stelle zurück, erhöhe den darin enthaltenen Wert um 1 und wiederhole diesen Punkt. Ist dies nicht möglich, da wir wieder am Anfang sind, so gibt es keine Inzidenzmatrix.

iv) Kommen wir mit diesem Algorithmus zum Ende der Matrix, so erhalten wir eine Inzidenzmatrix einer projektiven Ebene.

Auch ein Programm mit diesem Algorithmus wurde von M. Mann [19] erstellt, wodurch, zusammen mit einer geeigneten Einschränkung der ersten Permutationsmatrix, sich auch Ordnung 8 bewältigen ließ.

3 Eigenschaften des Kerns

Wir wollen uns nun mit den besonderen Eigenschaften der Permutationsmatrizen des Kerns, der Überdeckungsfreiheit und der Rechtecksregel, näher beschäftigen, insbesondere, wie sich diese unter bestimmten Umformungen verhalten.

3.1 Eigenschaften der Überdeckungsfreiheit

Definition 3.1
Im folgenden sei $\mathcal{S}_n$ die Menge der $(n \times n)$-Permutationsmatrizen. Da jede Permutationsmatrix $A \in \mathcal{S}_n$ auch als Permutation aufgefaßt werden kann (definiert durch $Ae_i = e_j \Leftrightarrow a(i) = j$), werden zur besseren Unterscheidung Permutationsmatrizen mit Großbuchstaben und die zugehörigen Permutationen mit den entsprechenden Kleinbuchstaben bezeichnet.
Für $A, B \in \{0,1\}^{m \times n}$ sei $C := A \wedge B$ durch $c_{i,j} := \begin{cases} 1 & \text{für } a_{i,j} = 1 = b_{i,j} \\ 0 & \text{sonst} \end{cases}$ definiert. Somit läßt sich die Überdeckungsfreiheit von A und B ausdrücken durch $A \wedge B = 0$.

Lemma 3.2
Seien $A, B \in \{0,1\}^{n \times m}$, $F \in \mathcal{S}_n$ und $G \in \mathcal{S}_m$. Es gelten folgende Rechenregeln:
$F(A \wedge B)G = FAG \wedge FBG$
$(A \wedge B)^\top = A^\top \wedge B^\top$.

Beweis:

Die Wirkung von F und G auf A läßt sich durch ein Vertauschen der Zeilen bzw. Spalten von A ausdrücken. Genauer gilt $(FAG)_{i,j} = (A)_{f^{-1}(i),g(j)}$. Für alle $i = 1, \ldots, n$ und $j \in 1, \ldots, m$ folgt damit:
$(F(A \wedge B)G)_{i,j} = 1 \Leftrightarrow (A \wedge B)_{f^{-1}(i),g(j)} = 1 \Leftrightarrow (A)_{f^{-1}(i),g(j)} = 1 = (B)_{f^{-1}(i),g(j)} \Leftrightarrow (FAG)_{i,j} = 1 = (FBG)_{i,j} \Leftrightarrow (FAG \wedge FBG)_{i,j} = 1$.
Es gilt weiterhin: $((A \wedge B)^\top)_{i,j} = 1 \Leftrightarrow (A \wedge B)_{j,i} = 1 \Leftrightarrow (A)_{j,i} = 1 = (B)_{j,i} \Leftrightarrow (A^\top)_{i,j} = 1 = (B^\top)_{i,j} \Leftrightarrow (A^\top \wedge B^\top)_{i,j} = 1$.

□

Korollar 3.3
Seien $A, B \in \{0,1\}^{n \times m}$, seien $F \in \mathcal{S}_n$ und $G \in \mathcal{S}_m$. Dann sind A und B genau dann überdeckungsfrei, wenn FAG und FBG überdeckungsfrei sind. Die Überdeckungsfreiheit bleibt also unter simultanem Permutieren der Zeilen und der Spalten erhalten.

Beweis:

Nach Lemma 3.2 gilt: A und B sind überdeckungsfrei $\Leftrightarrow A \wedge B = 0 \Leftrightarrow FAG \wedge FBG = F(A \wedge B)G = F0G = 0 \Leftrightarrow FAG$ und FBG sind überdeckungsfrei.

□

Korollar 3.4
Seien A, B und $F \in \mathcal{S}_n$. Dann sind A und B genau dann überdeckungsfrei, wenn FAF^{-1} und FBF^{-1} überdeckungsfrei sind. Die Überdeckungsfreiheit der Permutationsmatrizen bleibt also unter Konjugation erhalten.

Beweis:

Folgt unmittelbar aus Korollar 3.3 mit $G = F^{-1}$.

□

Dieses unscheinbare Korollar ist besonders nützlich, wenn es darum geht, die inneren Blockzeilen einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene, welche durch ein flächendeckendes Ensemble gegeben sind, zusammenzufassen. Hierbei ist die erste Permutationsmatrix als Einheitsmatrix festgelegt. Dazu können wir die zweite Matrix per Konjugation auf eine je Zyklenstruktur zurückführen, wobei durch dieses Korollar gesichert ist, daß die Überdeckungsfreiheit erhalten bleibt. Nun jedoch noch einige weitere Spezialfälle der Überdeckungsfreiheit bei Permutationsmatrizen.

Lemma 3.5
Seien A und $B \in \mathcal{S}_n$. Dann gilt:

i) *AB^{-1} ist genau dann fixpunktfrei, wenn A und B überdeckungsfrei sind.*
ii) *$B^{-1}A$ ist genau dann fixpunktfrei, wenn A und B überdeckungsfrei sind.*
iii) *A und AB^{-1} sind genau dann überdeckungsfrei, wenn B fixpunktfrei ist.*
iv) *A und $B^{-1}A$ sind genau dann überdeckungsfrei, wenn B fixpunktfrei ist.*
v) *A und B sind genau dann überdeckungsfrei, wenn A^{-1} und B^{-1} überdeckungsfrei sind. Insbesondere ist A genau dann fixpunktfrei, wenn A^{-1} fixpunktfrei ist.*

Beweis:

Man beachte: A und B überdeckungsfrei $\Leftrightarrow A \wedge B = 0$. Weiterhin ist A fixpunktfrei $\Leftrightarrow A$ und E sind überdeckungsfrei $\Leftrightarrow A \wedge E = 0$, wobei E die Einheitsmatrix ist. Für eine Permutationsmatrix A gilt außerdem noch $A^{-1} = A^{\top}$.

i) Nach Korollar 3.3 sind genau dann A und B überdeckungsfrei, wenn AB^{-1} und $BB^{-1} = E$ überdeckungsfrei sind, also AB^{-1} fixpunktfrei.
ii) analog i).
iii) Nach Lemma 3.2 ist $A \wedge AB^{-1} = 0 \Leftrightarrow E \wedge B^{-1} = 0 \Leftrightarrow (E \wedge B^{-1})^{\top} = E^{\top} \wedge (B^{-1})^{\top} = E \wedge B = 0 \Leftrightarrow$ B ist fixpunktfrei.
iv) analog iii).
v) Nach Lemma 3.2 ist $A \wedge B = 0 \Leftrightarrow A^{-1} \wedge B^{-1} = A^{\top} \wedge B^{\top} = (A \wedge B)^{\top} = 0$ $\Leftrightarrow A^{-1}$ und B^{-1} überdeckungsfrei. Der Rest folgt mit $B = E$.

□

Eine besondere Folgerung ergibt sich für Untergruppen von $\mathcal{S}_n$:

Korollar 3.6
Eine Untergruppe G von $\mathcal{S}_n$ ist genau dann überdeckungsfrei, wenn die Einheitsmatrix E als einzige Permutationsmatrix von G einen Fixpunkt enthält. In diesem Fall ist die Gruppenordnung $|G|$ ein Teiler von n und die Zyklen einer Permutationsmatrix aus G haben jeweils gleiche Länge.

Beweis:

Nach Lemma 3.5 i) sind zwei Permutationsmatrizen $A, B \in G$ genau dann überdeckungsfrei, wenn $AB^{-1} \in G$ fixpunktfrei ist, woraus die erste Aussage folgt. Betrachten wir nun die Wirkung der Gruppe G auf den Einheitsvektoren $e_1, \ldots, e_n$, so haben die Bahnen aufgrund der Überdeckungsfreiheit alle Länge $|G|$, es gilt somit $|G| \mid n$. Ist nun $P \in G$ eine Permutationsmatrix, welche einen l-Zyklus und einen m-Zyklus enthält mit $l < m$, so ist $P^l \in G$ und P^l enthält einen Fixpunkt aufgrund des l-Zyklus, doch P^l ist nicht die Einheitsmatrix aufgrund des m-Zyklus, im Widerspruch zur Überdeckungsfreiheit von G. □

Eine weitere interessante Eigenschaft ergibt sich beim Zentralisator eines flächendeckenden Ensembles. Dieser kann nämlich nur bestimmte Ordnungen annehmen. Genauer gilt:

Lemma 3.7
Seien $P_1, \ldots, P_n \in \mathcal{S}_n$ ein flächendeckendes Ensemble. Dann ist der Zentralisator $Z := C_{\mathcal{S}_n}(P_1, \ldots, P_n)$ überdeckungsfrei und $|Z|$ ist ein Teiler von n, insbesondere ist $|Z| \leq n$. Weiterhin ist $C_{\mathcal{S}_n}(P_1, \ldots, P_{n-1}) = C_{\mathcal{S}_n}(P_1, \ldots, P_n)$.

Beweis:

Da der Zentralisator eine Gruppe bildet, müssen wir nach Korollar 3.6 lediglich zeigen, daß E_n die einzige Permutationsmatrix in $Z := C_{\mathcal{S}_n}(P_1, \ldots, P_n)$ ist, welche einen Fixpunkt hat. Nehmen wir also an, $F \in C_{\mathcal{S}_n}(P_1, \ldots, P_n)$ hat einen Fixpunkt, also o. B. d. A. $Fe_1 = e_1$. Da $(P_i)_{1 \leq i \leq n}$ ein flächendeckendes Ensemble bildet, gibt es nach Lemma 2.12 zu jedem $j = 1, \ldots, n$ ein i_j, so daß $P_{i_j} e_1 = e_j$. Dann ist $Fe_j = (P_{i_j} F P_{i_j}^{-1}) e_j = P_{i_j} F e_1 = P_{i_j} e_1 = e_j$. Weil dies für alle $j = 1, \ldots, n$ gilt, ist $F = E_n$ und der Zentralisator somit überdeckungsfrei. Nach Korollar 3.6 ist damit jedoch die Gruppenordnung $|Z|$ ein Teiler von n, also insbesondere $|Z| \leq n$. Für die letzte Aussage sei $F \in S_{\mathcal{S}_n}(P_1, \ldots, P_{n-1})$. Da $(P_i)_{1 \leq i \leq n}$ ein flächendeckendes Ensemble bilden, ist $P_n = J - \sum_{i=1}^{n-1} P_i$ mit der Einsmatrix J. Dann ist jedoch $FP_nF^{-1} = F(J - \sum_{i=1}^{n-1} P_i)F^{-1} = FJF^{-1} - \sum_{i=1}^{n-1} FP_iF^{-1} = J - \sum_{i=1}^{n-1} P_i = P_n$. Somit ist auch $F \in C_{\mathcal{S}_n}(P_1, \ldots, P_n)$, also $C_{\mathcal{S}_n}(P_1, \ldots, P_{n-1}) = C_{\mathcal{S}_n}(P_1, \ldots, P_n)$.
□

Dieses Lemma gilt insbesondere für die Permutationsmatrizen einer inneren Blocklinie einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene, bilden diese doch ein flächendeckendes Ensemble. Eine besondere Folgerung ergibt sich noch für den Fall, daß Z maximal ist.

Korollar 3.8
Sei $M := \{P_1, \ldots, P_n\} \subseteq \mathcal{S}_n$ ein flächendeckendes Ensemble und $Z := C_{\mathcal{S}_n}(M)$ der Zentralisator. Ist $|Z| = n$, so ist $M = C_{\mathcal{S}_n}(Z)$, insbesondere also M eine Gruppe.

Beweis:

Nach Lemma 3.7 ist Z überdeckungsfrei. Im Falle $|Z| = n$ bildet Z daher ein flächendeckendes Ensemble. Nach Lemma 3.7 ist damit auch $|C_{\mathcal{S}_n}(Z)| \leq n$. Da $M \subseteq C_{\mathcal{S}_n}(Z)$, folgt aus Ordnungsgründen die Gleichheit und M ist insbesondere eine Gruppe.

□

3.2 Die Rechtecksregel

Nachdem wir uns im letzten Abschnitt mit der Überdeckungsfreiheit beschäftigt haben, behandeln wir in diesem Abschnitt die Rechtecksregel. Hierzu betrachten wir einen Block der Form $\begin{pmatrix} Q & R \\ P & X \end{pmatrix}$, wobei P, Q und $R \in \mathcal{S}_n$ und X durch P, Q und R derart definiert ist, daß X an genau den Stellen eine 1 hat, welche durch die Rechtecksregel verboten sind und an den restlichen Stellen 0. In diesem Fall bezeichnen wir das Quadrupel $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ als **Rechteckskonfiguration** und die Menge aller Rechteckskonfigurationen mit $\mathcal{R}$. Man kann X aus den anderen Matrizen recht einfach nach dem folgenden Verfahren konstruieren:

$$\begin{array}{c} \begin{matrix} \overset{k}{\downarrow} & & & \overset{l}{\downarrow} \end{matrix} \\ \left(\begin{array}{ccc|ccc} 1 & \cdots & & \cdots & 1 & \\ \vdots & & & & \vdots & \\ \hline \vdots & & & & \vdots & \\ 1 & \cdots & & \cdots & 1 & \end{array}\right) \begin{matrix} \\ \\ \leftarrow j \\ \\ \\ \leftarrow i \end{matrix} \end{array} \tag{1}$$

Für die i-te Zeile von X suchen wir zunächst in der i-ten Zeile von P die Spalte k, in der P eine 1 als Eintrag hat. Eine solche Spalte existiert und ist eindeutig, da P eine Permutationsmatrix ist. Zu der Spalte k von Q bestimmen wir nun die Zeile j, in der Q eine 1 stehen hat. Diese ist wieder eindeutig, da auch Q eine Permutationsmatrix ist. Als nächstes suchen wir noch in der j-ten Zeile von R die Spalte l, die eine 1 enthält. Wiederum ist diese eindeutig, da auch R eine Permutationsmatrix ist. Nun müssen wir nur

noch in der i-ten Zeile von X in der l-ten Spalte eine 1 eintragen und die restliche Zeile mit 0 auffüllen. Wendet man diese Verfahren für jede Zeile von X an, hat man am Ende X aus den restlichen Matrizen bestimmt. Dieses Verfahren führt immer zu einer eindeutig durch P, Q und R bestimmten Matrix, so daß wir $X = f(P, Q, R)$ als Funktion von $P,\ Q$ und R betrachten können. Weiterhin ist X sogar eine Permutationsmatrix, daran zu erkennen, daß X durch Vertauschen der Zeilen von R entsteht.

Satz 3.9
$\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ *ist genau dann eine Rechteckskonfiguration, wenn* $X = PQ^{-1}R$ *ist.*

Beweis:

Betrachten wir ein Rechteck mit derselben Indizierung wie in (1). Dann gilt $Xe_l = e_i = Pe_k$ und $Re_l = e_j = Qe_k$. Die zweite Gleichung läßt sich umschreiben in $e_k = Q^{-1}Re_l$. Dies in die erste Gleichung eingesetzt ergibt $Xe_l = Pe_k = P(Q^{-1}Re_l) = PQ^{-1}Re_l$. Da dies für alle Standard-Einheitsvektoren e_l gilt, folgt daraus $X = PQ^{-1}R$.

□

Für den Spezialfall $Q = E = R$ erhalten wir als verbotene Stellen $X = P$, d. h. die an der Stelle von X bezüglich der Rechtecksregel erlaubten Matrizen sind gerade die mit P überdeckungsfreien Matrizen. Dies zeigt somit noch einmal, daß die Überdeckungsfreiheit gerade die Anwendung der Rechtecksregel mit der 0-ten Blockzeile ist. Als Nächstes untersuchen wir verschiedene Operationen, unter denen eine Rechteckskonfiguration erhalten bleibt. Vertauschen der Blocklinien und Transponieren führt zu folgendem Satz:

Satz 3.10
Folgende Aussagen sind äquivalent:

i)	$\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R}$	v)	$\begin{bmatrix} Q^{-1} & P^{-1} \\ R^{-1} & X^{-1} \end{bmatrix} \in \mathcal{R}$
ii)	$\begin{bmatrix} P & X \\ Q & R \end{bmatrix} \in \mathcal{R}$	vi)	$\begin{bmatrix} R^{-1} & X^{-1} \\ Q^{-1} & P^{-1} \end{bmatrix} \in \mathcal{R}$
iii)	$\begin{bmatrix} R & Q \\ X & P \end{bmatrix} \in \mathcal{R}$	vii)	$\begin{bmatrix} P^{-1} & Q^{-1} \\ X^{-1} & R^{-1} \end{bmatrix} \in \mathcal{R}$
iv)	$\begin{bmatrix} X & P \\ R & Q \end{bmatrix} \in \mathcal{R}$	viii)	$\begin{bmatrix} X^{-1} & R^{-1} \\ P^{-1} & Q^{-1} \end{bmatrix} \in \mathcal{R}$

Beweis:
ii): $\begin{bmatrix} P & X \\ Q & R \end{bmatrix} \in \mathcal{R} \Leftrightarrow R = QP^{-1}X \Leftrightarrow X = (PQ^{-1})(QP^{-1}X) = PQ^{-1}R \Leftrightarrow$ i): $\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R} \Leftrightarrow P = (PQ^{-1}R)(R^{-1}Q) = XR^{-1}Q \Leftrightarrow$ iii): $\begin{bmatrix} R & Q \\ X & P \end{bmatrix} \in \mathcal{R}$. Weiterhin ist i): $\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R} \Leftrightarrow X = PQ^{-1}R \Leftrightarrow X^{-1} = (PQ^{-1}R)^{-1} = R^{-1}QP^{-1} \Leftrightarrow$ v): $\begin{bmatrix} Q^{-1} & P^{-1} \\ R^{-1} & X^{-1} \end{bmatrix} \in \mathcal{R}$. Die restlichen Rechteckskonfigurationen ergeben sich durch Hintereinanderausführung dieser drei Transformationen.

□

Eine weitere Umformung, welche ein Rechteck wieder auf ein Rechteck abbildet, ist eine Zeilenvertauschung in einer der Blockzeilen oder eine Spaltenvertauschung in einer der Blockspalten. Dies führt zu folgendem Satz:

Satz 3.11
Sind A, B, C, $D \in \mathcal{S}_n$, so ist genau dann $\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R}$, *wenn*
$\begin{bmatrix} AQC & ARD \\ BPC & BXD \end{bmatrix} \in \mathcal{R}$.

Beweis:
$\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R} \Leftrightarrow X = PQ^{-1}R \Leftrightarrow BXD = B(PQ^{-1}R)D = BP(CC^{-1})Q^{-1}(A^{-1}A)RD = (BPC)(AQC)^{-1}(ARD) \Leftrightarrow \begin{bmatrix} AQC & ARD \\ BPC & BXD \end{bmatrix} \in \mathcal{R}$.

□

Setzt man $C = D = E$, die Einheitsmatrix, so bewirkt diese Transformation gerade ein Vertauschen der Zeilen, setzt man dagegen $A = B = E$, so werden durch diese Transformation gerade die Spalten vertauscht. Im nächsten Satz betrachten wir eine komplette Drehung der Matrix um 90°. Da hierbei Rechtecke auf Rechtecke abgebildet werden, ist es einleuchtend, daß auch die gedrehte Matrix eine Rechteckskonfiguration ist. Zur einfacheren Schreibweise bezeichnen wir mit A° die um 90° nach rechts gedrehte Matrix A.

Satz 3.12
$\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R} \quad \Leftrightarrow \quad \begin{bmatrix} P^\circ & Q^\circ \\ X^\circ & R^\circ \end{bmatrix} \in \mathcal{R}$.

Zunächst sei bemerkt, daß ein Übergang von P zu P° ein Übergang von $(P)_{i,j}$ zu $(P^\circ)_{i,j} = (P)_{j,n+1-i}$ ist. Nun ist $(P)_{j,i} = (P^\top)_{i,j}$ und $(P)_{j,n+1-i} = (PK)_{j,i}$, wobei $K = \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix}$. Dies ergibt zusammen $P^\circ = KP^\top$.

Beweis:
Nach Satz 3.10 vii) ist genau dann $\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R}$, wenn $\begin{bmatrix} P^{-1} & Q^{-1} \\ X^{-1} & R^{-1} \end{bmatrix} \in \mathcal{R}$, und dies ist nach Satz 3.11 (mit $A = B = K,\ C = D = E$) genau dann der Fall, wenn $\begin{bmatrix} KP^{-1} & KQ^{-1} \\ KX^{-1} & KR^{-1} \end{bmatrix} = \begin{bmatrix} P^\circ & Q^\circ \\ X^\circ & R^\circ \end{bmatrix} \in \mathcal{R}$.

□

Die letzten drei Sätze behandelten zahlreiche gültige Transformationen einer Rechteckskonfiguration. Hierbei stellt sich die Frage, wie sich die ,,offensichtlichen“ Symmetrien, wie Drehungen um Vielfache von 90° und Spiegelungen an einer horizontalen, vertikalen oder auch diagonalen Achse, in dieses Bild einfügen lassen.
Die Drehungen sind durch den letzten Satz behandelt. Die Spiegelung an der horizontalen Achse dagegen läßt sich durch Vertauschen der oberen mit den unteren Blöcken und gleichzeitig einem geeigneten Permutieren der Zeilen erreichen. Genauer können wir dies schreiben als Übergang der Matrix $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ zu $\begin{bmatrix} KP & KX \\ KQ & KR \end{bmatrix}$ mit K wie oben. Dies ist nach Satz 3.10 ii) und Satz 3.11 eine mit der Rechtecksregel verträgliche Transformation. Ähnlich erreichen wir die Spiegelung an der vertikalen Achse durch einen nach Satz 3.10 iii) und Satz 3.11 ebenfalls mit der Rechtecksregel verträglichen Übergang von $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ zu $\begin{bmatrix} RK & QK \\ XK & PK \end{bmatrix}$. Die Spiegelung an der einen Diagonalen wiederum ist durch das Transponieren in Satz 3.10 v) behandelt worden, die Spiegelung an der anderen Diagonalen läßt sich durch geeignete Kombination der Transposition mit der Drehung um 90° verwirklichen. Nun jedoch ein Satz zur Überdeckungsfreiheit:

Satz 3.13
Sei $\begin{bmatrix} Q & R \\ P & X \end{bmatrix} \in \mathcal{R}$. *Dann gilt:*

i) *P und X sind genau dann überdeckungsfrei, wenn Q und R überdeckungsfrei sind.*

ii) *R und X sind genau dann überdeckungsfrei, wenn Q und P überdeckungsfrei sind.*

Beweis:

zu i):
Nach Lemma 3.5 ii) sind Q und R genau dann überdeckungsfrei, wenn $R^{-1}Q$ fixpunktfrei ist. Dies ist nach Lemma 3.5 i) genau dann der Fall, wenn P und $P(R^{-1}Q)^{-1} = PQ^{-1}R = X$ überdeckungsfrei sind.
zu ii):
Nach Lemma 3.5 i) sind Q und P genau dann überdeckungsfrei, wenn QP^{-1} fixpunktfrei ist. Dies ist nach Lemma 3.5 ii) genau dann der Fall, wenn R und $(QP^{-1})^{-1}R = PQ^{-1}R = X$ überdeckungsfrei sind.

□

Dieser Satz zeigt in gewisser Weise die Effektivität der Rechtecksregel. Da unser Ziel ein Algorithmus ist, der bei einer schon gegebenen Blockzeile des Permutationsteils eine weitere Blockzeile unter Wahrung der Fixpunktfreiheit, der Überdeckungsfreiheit und der Rechtecksregel anfügt, ist es natürlich von Vorteil, wenn durch jede dieser Regeln verschiedene Stellen verboten sind, so daß die Anzahl der an der neuen Stelle widerspruchsfrei setzbaren Matrizen möglichst gering ist. Nun sind die bei gegebenen P, Q und R die für X unter Wahrung der Rechtecksregel einsetzbaren Matrizen durch die Überdeckungsfreiheit mit P und mit R eingeschränkt, aber auch mit $PQ^{-1}R$ durch die Rechtecksregel. Diese bedingt nach diesem Satz ausschließlich echte, zur Überdeckungsfreiheit zusätzliche Einschränkungen. Nun wäre die Vermutung naheliegend, und für den Algorithmus auch wünschenswert, eine ähnliche Aussage über die Fixpunktfreiheit von X machen zu können. Dies ist im Allgemeinen jedoch falsch, wie folgendes Beispiel zeigt. Seien Q, R und $P \in \mathcal{S}_3$, die zugehörigen Permutationen seien $q = (132)$ und $p = r = (123)$. Dann sind Q, R und P fixpunktfrei und sowohl Q und R also auch Q und P sind jeweils überdeckungsfrei. Dennoch ist $x = pq^{-1}r = (123) \cdot (132)^{-1} \cdot (123) = (123) \cdot (123) \cdot (123) = \mathrm{id}$, also X nicht fixpunktfrei.

4 Lateinische Quadrate

4.1 Grundlagen

Definition 4.1
Ein **lateinisches Quadrat der Ordnung** n ist eine $(n \times n)$-Matrix mit Einträgen aus $\{1, \ldots, n\}$, so daß jede Zahl in jeder Zeile genau einmal vorkommt und jede Zahl in jeder Spalte genau einmal vorkommt. Zwei lateinische Quadrate A und B gleicher Ordnung heißen **orthogonal**, wenn in der Matrix $C := (A, B)$, definiert durch $c_{i,j} := (a_{i,j}, b_{i,j})$, jedes Paar nur einmal vorkommt.

Beispiel 4.2
Die beiden Matrizen

$$A_1 := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{pmatrix} \text{ und } A_2 := \begin{pmatrix} 1 & 3 & 4 & 2 \\ 2 & 4 & 3 & 1 \\ 3 & 1 & 2 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}$$

sind ein Paar von orthogonalen lateinischen Quadraten der Ordnung 4, wie sich leicht nachprüfen läßt. Lateinische Quadrate lassen sich auf folgende Art konstruieren:
Sei $(G, \circ)$ eine Gruppe der Ordnung n. Wir identifizieren die Elemente von G mit den Zahlen $1, \ldots, n$. Die Multiplikationstafel oder **Cayley-Tafel** $A = (a_{i,j})_{1 \le i,j \le n}$, definiert durch $a_{i,j} = i \circ j$ für alle $i, j = 1, \ldots, n$, ist dann ein lateinisches Quadrat. Allgemeiner gilt: Die lateinischen Quadrate sind gerade die Cayley-Tafeln von Quasi-Gruppen. Eine **Quasi-Gruppe** ist hierbei ein Paar $(Q, \circ)$ mit einer Menge Q und einer Verknüpfung $\circ$ auf Q, so daß die Gleichungen $x \circ y = z$ und $y \circ x = z$ bei gegebenen $y, z \in Q$ immer eindeutig nach x auflösbar sind (vgl. Kochendörffer [14]).

In unserem Fall ist das lateinische Quadrat A_1 die Cayley-Tafel der kleinschen Vierergruppe $\mathbb{Z}_2 \times \mathbb{Z}_2$, wenn wir die Gruppenelemente mit den Zahlen $1, \ldots, 4$ durch $1 := (0,0)$, $2 := (1,0)$, $3 := (0,1)$ und $4 := (1,1)$ identifizieren.

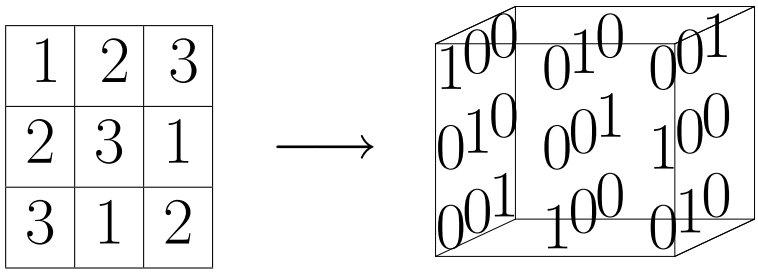
Sei $\mathbb{L}_n$ die Menge der lateinischen Quadrate der Ordnung n. Wir wollen zunächst Gruppen untersuchen, welche auf $\mathbb{L}_n$ wirken. Eine solche Gruppe ist die Gruppe U_1, welche aus den Permutationen der Zeilen besteht. Analog wirkt auch die Gruppe U_2, welche aus den Permutationen der Spalten besteht, auf $\mathbb{L}_n$. Als drittes betrachten wir noch die Gruppe U_3, welche aus den Umnummerierungen der Einträge des lateinischen Quadrates besteht. Für diese drei Gruppen gilt jeweils $U_i \cong S_n$. Nun vertauschen U_1 und U_2 elementweise. Dies sieht man am einfachsten, wenn man die Zeilenpermutation als Multiplikation des lateinischen Quadrates mit einer Permutationsmatrix von links, und die Spaltenpermutation als Multiplikation mit einer Permutationsmatrix von rechts betrachtet. Dann folgt die Vertauschbarkeit der

Elemente einfach aus der Assoziativität der Matrizenmultiplikation. Ebenso vertauschen U_3 mit U_1 und U_3 mit U_2 elementweise. Daher ist die Wirkung der Gruppe $U := U_1 \times U_2 \times U_3 \cong S_n \times S_n \times S_n$ auf $\mathbb{L}_n$ wohldefiniert. Diese Wirkung ist im Allgemeinen nicht treu, so bleiben beispielsweise die lateinischen Quadrate von $\mathbb{L}_2 = \{\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\}$ unter gleichzeitigem Permutieren der Zeilen und der Spalten erhalten. Die Bahnen von U heißen **Familien von lateinischen Quadraten** (vgl. Tarry [25]).

Als nächstes betrachten wir noch die Gruppe V, welche aus den Vertauschungen von Zeilen, Spalten und Nummerierung besteht. Hierbei ist das Vertauschen von Zeilen und Spalten gerade das Transponieren der Matrix, wodurch ein lateinisches Quadrat natürlich wieder in ein lateinisches Quadrat umgeformt wird. Um die Nummerierung und die Umformungen damit besser zu sehen, bringen wir das lateinische Quadrat zunächst in eine andere Form. Wir schreiben das lateinische Quadrat A der Ordnung n als $A = 1 \cdot P_1 + 2 \cdot P_2 + \cdots + n \cdot P_n$ mit einem flächendeckenden Ensemble von Permutationsmatrizen $P_1, \ldots, P_n$ wie im folgenden Beispiel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} = 1 \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + 2 \cdot \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 3 \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Diese Aufspaltung ist stets eindeutig. Schreiben wir die Permutationsmatrizen nun räumlich hintereinander, so erhalten wir einen $(0,1)$-Würfel wie folgt:



Die vertikale Koordinate des Würfels ist hierbei die Zeile, die horizontale die Spalte und die Koordinate in die Tiefe ist die Nummerierung. Auffällig ist hierbei, daß jede Ebene, welche zu einer der Seiten des Würfels parallel ist, eine Permutationsmatrix darstellt. Ist die Ebene zur Blattebene parallel, so folgt dies nach Konstruktion, ist die Ebene parallel zur Horizontalen oder Vertikalen, so folgt dies aus der Überdeckungsfreiheit der Permutationsmatrizen. Umgekehrt bilden bei einem $(0,1)$-Würfel mit der Eigenschaft, daß jede zu einer der Seiten parallele Ebene eine Permutationsmatrix darstellt, die parallelen Ebenen jeweils ein flächendeckendes Ensemble von Permutationsmatrizen. Nun bewirkt das Vertauschen von Zeilen und Spalten, also Transponieren, eines lateinischen Quadrates genau das Spiegeln des zugehörigen $(0,1)$-Würfels an der zur Blattebene senkrechten Diagonalen in

NW-Richtung. Analog bewirkt das Vertauschen von Zeilen und Nummerierung bzw. von Spalten und Nummerierung genau das Spiegeln des zugehörigen $(0,1)$-Würfels an entsprechenden Diagonalen senkrecht zur vertikalen bzw. horizontalen Ebene, wodurch natürlich auch weiterhin die parallelen Ebenen ein vollständiges Ensemble von Permutationsmatrizen bilden. Also führen auch diese Vertauschungen lateinische Quadrate wieder in lateinische Quadrate über. Allerdings ist die Wirkung von V ebenfalls im Allgemeinen nicht treu, wie man wiederum am Beispiel von $\mathbb{L}_2$ sieht.

Nun wollen wir die Beziehungen von U und V zueinander etwas näher untersuchen. Sei hierzu $\vartheta_i : S_n \to U_i$ der kanonische Isomorphismus für $i = 1, 2, 3$. Da V die drei Komponenten Zeilen, Spalten und Nummerierung permutiert, ist $V \cong S_3$, weswegen wir im folgenden auch V mit S_3 identifizieren. Sind nun $p \in V$ und $u \in U$, so gibt es drei Permutationen α, β, $\gamma \in S_n$ mit $u = \vartheta_1(\alpha)\vartheta_2(\beta)\vartheta_3(\gamma)$. Dann ist jedoch $pup^{-1} = p\vartheta_1(\alpha)\vartheta_2(\beta)\vartheta_3(\gamma)p^{-1} = \left(p\vartheta_1(\alpha)p^{-1}\right)\left(p\vartheta_2(\beta)p^{-1}\right)\left(p\vartheta_3(\gamma)p^{-1}\right) = \vartheta_{p(1)}(\alpha)\vartheta_{p(2)}(\beta)\vartheta_{p(3)}(\gamma) \in U$. Das heißt jedoch gerade, daß U normal bezüglich V ist. Es ist daher die Gruppe $T := U \underset{\varphi}{\rtimes} V \cong (S_n \times S_n \times S_n) \underset{\varphi}{\rtimes} S_3$ als semidirektes Produkt wohldefiniert und wirkt auf $\mathbb{L}_n$. Dabei ist die Wirkung von φ durch $\varphi(p)\left(\vartheta_1(\alpha)\vartheta_2(\beta)\vartheta_3(\gamma)\right) = \vartheta_{p(1)}(\alpha)\vartheta_{p(2)}(\beta)\vartheta_{p(3)}(\gamma)$ für $p \in V$, $\alpha, \beta, \gamma \in S_n$ gegeben. Die Bahnen von T auf $\mathbb{L}_n$ heißen **Isomorphieklassen** (vgl. Betten [2]).

Die „offensichtlichen“ Transformationen wie Spiegelungen und Drehungen des Würfels sind dabei schon in T enthalten. So erreicht man die Spiegelungen an einer zu einer der Flächen des Würfels parallelen Ebene durch geeignete Permutation der Zeilen bzw. Spalten bzw. Nummerierung. Die Spiegelung an der zur Blattebene senkrechten Diagonalen in NW-Richtung erreicht man durch Vertauschen von Zeilen und Spalten, die Spiegelung an der anderen zur Blattebene senkrechten Diagonalen, der in NO-Richtung, erreicht man durch Konjugation dieser Spiegelung mit der Spiegelung an der horizontalen, zur Blattebene senkrechten Ebene. In ähnlicher Weise lassen sich die Spiegelungen an den anderen diagonalen Ebenen des Würfels durch Operationen aus T verwirklichen. Die Drehung des Würfels um die zur Blattebene senkrechten Achse um 90° läßt sich durch Kombination der Spiegelung an der zur Blattebene senkrechten diagonalen Ebene in NO-Richtung mit der Spiegelung an der zur Blattebene senkrechten, vertikalen Ebene erzeugen. In analoger Weise können wir auch die Drehungen an den anderen zu einer der Flächen des Würfels senkrechten Achsen verwirklichen. Bleiben noch die Drehungen um eine der diagonalen Achsen des Würfels. Die Drehungen um die Achse durch die vordere, linke obere Ecke und die hintere, rechte untere Ecke erreichen wir durch Vertauschen der Komponenten Zeile, Spalte und Nummerierung. Die Drehungen um eine der anderen diagonalen Achsen bekommen wir dann durch Konjugation dieser Drehung mit einer geeigneten Drehung um eine zu einer Fläche des Würfels senkrechten Achse.

Die in Beispiel 4.2 vorgestellten Cayley-Tafeln von Gruppen, oder allgemeiner von Quasi-Gruppen, sind im Allgemeinen nicht eindeutig durch den Isomorphietyp der Gruppe bestimmt, sondern können sich mit den verschiedenen Identifikationen der Gruppenelemente mit den Zahlen $1, \ldots, n$ ändern. Diese Änderungen bewirken jedoch bei den erzeugten lateinischen Quadraten eine Kombination aus Zeilenvertauschungen, Spaltenvertauschungen und Umnummerierungen, so daß zwar nicht die Cayley-Tafel selbst, doch zumindest die Isomorphieklasse der Cayley-Tafel durch den Isomorphietyp der Gruppe eindeutig bestimmt ist. Wir sprechen daher in Zukunft von ,,der Cayley-Tafel" einer Gruppe, auch wenn nur ihre Isomorphieklasse durch die Gruppe festgelegt ist.

Die durch T gegebenen Operationen erlauben es uns weiterhin, die lateinischen Quadrate in gewisser Weise zu normieren. Zwei orthogonale lateinische Quadrate A und B bleiben orthogonal, wenn wir die Zeilen oder die Spalten simultan bei A und B vertauschen. Ebenso bleiben sie orthogonal, wenn wir bei B die Nummerierung vertauschen. Durch diese Operationen lassen sich zwei orthogonale lateinische Quadrate A und B immer umformen, so daß

$$A = \begin{pmatrix} 1 & 2 & \cdots & n \\ 2 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ n & * & \cdots & * \end{pmatrix} \text{ und } B = \begin{pmatrix} 1 & * & \cdots & * \\ 2 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ n & * & \cdots & * \end{pmatrix}.$$

Sind $A^{(1)}, \ldots, A^{(m)}$ ein System von paarweisen orthogonalen lateinischen Quadraten, so ist $m \leq n-1$. Dies sieht man am einfachsten, wenn man sich die lateinischen Quadrate wie oben normiert bezüglich der ersten Spalte vorstellt. Dann kann bei $(A^{(k)})_{1,2}$ jede Zahl außer 1 höchstens einmal vorkommen für alle $k = 1, \ldots, m$. Somit ist $m \leq n-1$ (vgl. Brualdi, Ryser [7]). Wir sprechen daher im Falle $m = n-1$ auch von einem **vollständigen System orthogonaler lateinischer Quadrate**.

4.2 Lateinische Quadrate und projektive Ebenen

In diesem Abschnitt werden wir eine interessante Beziehung zwischen den lateinischen Quadraten und dem Permutationsteil einer doppelgeordneten Inzidenzmatrix herleiten. Betrachten wir hierzu eine innere Blockzeile. Sehen wir von den Beziehungen der inneren Blockzeilen untereinander ab, so gilt als einzige Bedingung für eine gültige innere Blockzeile die Überdeckungsfreiheit der verschiedenen Permutationsmatrizen, denn die Bedingung der Fixpunktfreiheit des Permutationsteils läßt sich als Überdeckungsfreiheit mit der Einheitsmatrix in der 0-ten Blockspalte auffassen. Betrachten wir nun die einzelnen Permutationsmatrizen $F^{(1)}, \ldots, F^{(n)}$ der Blockzeile als Permutationen $f^{(1)}, \ldots, f^{(n)}$ auf $\{1, \ldots, n\}$, definiert durch $f^{(k)}(i) = j \Leftrightarrow$

$F^{(k)}e_i = e_j$ für $i, j, k \in 1, \ldots, n$ und schreiben die Matrix

$$A := \begin{pmatrix} f^{(1)}(1) & f^{(2)}(1) & \cdots & f^{(n)}(1) \\ f^{(1)}(2) & f^{(2)}(2) & \cdots & f^{(n)}(2) \\ \vdots & \vdots & \ddots & \vdots \\ f^{(1)}(n) & f^{(2)}(n) & \cdots & f^{(n)}(n) \end{pmatrix}.$$

Dann ist A ein lateinisches Quadrat, denn in jeder Spalte kommt jede Zahl genau einmal vor, da die $f^{(i)}$ Permutationen sind, und in jeder Zeile kommt jede Zahl genau einmal vor, da die $f^{(i)}$ überdeckungsfrei sind. Weil dies genaue Entsprechungen sind, gilt auch die Umkehrung, also jedes lateinische Quadrat läßt sich nach dieser Methode in eine gültige innere Blockzeile übersetzen. Analoges gilt natürlich auch für die inneren Blockspalten. Betrachten wir als kleines Beispiel (für $n = 3$) die folgende innere Blockzeile und das dazugehörige lateinische Quadrat:

$$\left(\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right) \quad \longrightarrow \quad \begin{pmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

Man beachte, daß durch die Einheitsmatrix der 0-ten Blockspalte die erste Zeile des lateinischen Quadrates normiert ist. Schreibt man umgekehrt ein lateinisches Quadrat in eine Blockzeile um, so ist natürlich die erste Permutationsmatrix im Allgemeinen keine Einheitsmatrix, die Blockzeile ist dennoch eine gültige innere Blockzeile, jedoch ist die zugehörige Inzidenzmatrix natürlich nicht mehr vollständig doppelgeordnet.

Als nächstes betrachten wir, ähnlich der Rechtecksregel, die Bedingungen für die Blockzeilen, wenn die zugehörigen lateinischen Quadrate orthogonal sind. Hierzu seien A und B zwei orthogonale lateinische Quadrate, die p-te Spalte von A übersetzen wir in die Permutationsmatrix Q, die p-te Spalte von B in die Permutationsmatrix P, die q-te Spalte von A in die Permutationsmatrix R und X sei diejenige Permutationsmatrix, welche sich nach unserer Methode in eine q-te Spalte von B übersetzen ließe, welche durch die Orthogonalitätsregel bezüglich der p-ten und q-ten Spalten von A und B vollständig verboten wäre. In diesem Fall bezeichnen wir das Quadrupel $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ eine **Orthogonalitätskonfiguration**.

Satz 4.3
Es ist $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ *genau dann eine Orthogonalitätskonfiguration, wenn* $X = PQ^{-1}R$.

Beweis:

Die Orthogonalitätsregel besagt: Sei $i \in \{1, \ldots, n\}$ beliebig und sei $A_{i,q} := a$, übersetzt auf die Permutationsmatrizen heißt dies $Re_i = e_a$. Dann gibt es ein $j \in \{1, \ldots, n\}$ mit $A_{j,p} = a$, also $Qe_j = e_a$. Sei dazu $b = B_{j,p}$, also $Pe_j = e_b$. Dann ist nach der Orthogonalitätsregel $B_{i,q} = b$ verboten, das auf die Permutationsmatrizen übersetzt $Xe_i = e_b$ bedeutet. Wir haben also die beiden Gleichungen $Re_i = e_a = Qe_j$ und $Pe_j = e_b = Xe_i$. Die erste Gleichung läßt sich umschreiben in $e_j = Q^{-1}Re_i$, und dies in die zweite Gleichung eingesetzt ergibt $Xe_i = PQ^{-1}Re_i$. Da dies für alle $i = 1, \ldots, n$ gilt, folgt hieraus $X = PQ^{-1}R$.

□

Korollar 4.4

Es ist $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ *genau dann eine Orthogonalitätskonfiguration, wenn* $\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ *eine Rechteckskonfiguration ist.*

Beweis:

$\begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ ist Orthogonalitätskonfiguration $\Leftrightarrow X = PQ^{-1}R \Leftrightarrow \begin{bmatrix} Q & R \\ P & X \end{bmatrix}$ ist Rechteckskonfiguration nach Satz 4.3 und Satz 3.9

□

Die Beziehung zweier innerer Blockzeilen untereinander ist bestimmt durch die Gültigkeit der Rechtecksregel auf den Blöcken des Permutationsteils und der Überdeckungsfreiheit der Blöcke des Permutationsteils, welche in derselben Blockspalte sind. Diese Überdeckungsfreiheit läßt sich jedoch gerade als Anwendung der Rechtecksregel mit den Einheitsmatrizen der 0-ten Blockspalte interpretieren. Die einzige Beziehung der inneren Blockzeilen ist somit die Rechtecksregel, welche nach vorigem Korollar äquivalent zu der Orthogonalität der zugehörigen lateinischen Quadrate ist. Dies führt zu folgendem Korollar:

Korollar 4.5

Es gibt genau dann ein vollständiges System orthogonaler lateinischer Quadrate der Ordnung n, wenn es eine projektive Ebene der Ordnung n gibt.

Beweis:

Schreibt man das vollständige System normierter orthogonaler lateinischer Quadrate nach obigen Schema um, in die $n - 1$ inneren Blockzeilen, so erfüllen sie nach Korollar 4.4 auch die Rechtecksregel, die erhaltene gesamte Matrix ist also die Inzidenzmatrix einer projektiven Ebene der Ordnung n. Umgekehrt bilden die umgeschriebenen $n - 1$ inneren Blockzeilen der doppelgeordneten Inzidenzmatrix ein vollständiges System orthogonaler lateinischer Quadrate, ebenfalls nach Korollar 4.4.

□

Dieser Sachverhalt wurde bereits 1938 von Bose [4] bewiesen. Das schöne an dem Zugang über die doppelgeordnete Inzidenzmatrix ist jedoch die einfache und direkte Übersetzung der Inzidenzmatrix in das vollständige System orthogonaler lateinischer Quadrate und umgekehrt.

4.3 Umformungen doppelgeordneter Inzidenzmatrizen

Bei unserer Beschreibung des Permutationsteils durch lateinische Quadrate haben wir die Permutationsmatrizen als Spaltenmatrizen aufgefaßt. Man kann die Permutationsmatrizen jedoch auch als Zeilenmatrizen auffassen. Dann lassen sich zu einer inneren Blockzeile $F^{(1)}, \ldots, F^{(n)} \in \mathcal{S}_n$ die Permutationen $\tilde{f}^{(1)}, \ldots, \tilde{f}^{(n)}$, definiert durch $\tilde{f}^{(k)}(i) = j \Leftrightarrow e_i^\top F^{(k)} = e_j^\top$, bestimmen. Analog schreiben wir die Matrix

$$\tilde{A} := \begin{pmatrix} \tilde{f}^{(1)}(1) & \tilde{f}^{(2)}(1) & \cdots & \tilde{f}^{(n)}(1) \\ \tilde{f}^{(1)}(2) & \tilde{f}^{(2)}(2) & \cdots & \tilde{f}^{(n)}(2) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{f}^{(1)}(n) & \tilde{f}^{(2)}(n) & \cdots & \tilde{f}^{(n)}(n) \end{pmatrix}.$$

Auch $\tilde{A}$ ist dann ein lateinisches Quadrat, denn in jeder Spalte kommt jede Zahl genau einmal vor, da die $F^{(k)}$ Permutationsmatrizen sind, und in jeder Zeile kommt jede Zahl genau einmal vor, da die $F^{(k)}$ überdeckungsfrei sind. Eine äquivalente Definition der $\tilde{f}^{(k)}$ ist $\tilde{f}^{(k)}(i) = j \Leftrightarrow$ in der i-ten Zeile von $F^{(k)}$ steht der Einheitsvektor $e_j^\top$. Analog ist eine äquivalente Definition der $f^{(k)}$ unseres ursprünglichen lateinischen Quadrates $f^{(k)}(i) = j \Leftrightarrow$ in der i-ten Spalte von $F^{(k)}$ steht der Einheitsvektor e_j. Es unterscheiden sich $\tilde{f}^{(k)}$ und $f^{(k)}$ also nur durch Vertauschung von Zeile und Spalte der Permutationsmatrix $F^{(k)}$. Dies wiederum ist gerade das Transponieren von $F^{(k)}$, und da $F^{(k)}$ eine Permutationsmatrix ist, bedeutet das wiederum genau das Invertieren der Matrix. Es ist somit $\tilde{f}^{(k)} = \left(f^{(k)}\right)^{-1}$. Die neue Transformation einer inneren Blockzeile in ein lateinisches Quadrat hat jedoch technische Vorteile für unseren späteren Algorithmus, da sich ein zeilenweises Vorgehen, welches sich hier anbietet, besser mit dieser Transformation verträgt. Daher gehen wir künftig immer von dieser Umformung aus, falls nicht anders angegeben. Ein weiterer schöner Nebeneffekt ist, daß wir den zu dem lateinischen Quadrat gehörenden $(0,1)$-Würfel vom Anfang des Kapitels aus der Blockzeile ganz einfach erhalten, indem wir jede Permutationsmatrix um die vertikale Achse nach hinten drehen und dann die ganzen Permutationsmatrizen nebeneinander stellen. Hierbei sieht man, daß die relativen Zeilen des Permutationsteils gerade die Zeilen der lateinischen Quadrate sind, die Blockspalten sind die Spalten der lateinischen Quadrate und die relativen Spalten des Permutationsteils übersetzen sich in die Nummerierung der lateinischen Quadrate. Ein Nachteil bei dieser Interpretationsweise ist, daß

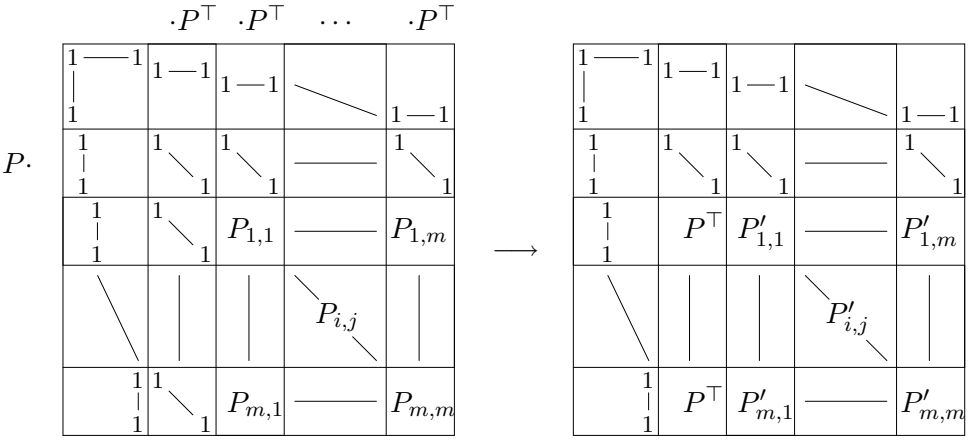
zwei transformierte Blockzeilen A und B einer Inzidenzmatrix nun im Allgemeinen nicht mehr orthogonal sind, stattdessen gilt die Rechtecksregel: ist $a_{i,j} = b_{k,j}$, so ist $a_{i,l} \neq b_{k,l}$ für alle $i,\ j,\ k,\ l \in \{1, \ldots, n\},\ l \neq j$.

Nun ist uns weniger an den Inzidenzmatrizen selbst, als vielmehr an den projektiven Ebenen, die durch diese beschrieben werden, gelegen. Die zugrundeliegende Geometrie bleibt jedoch unter Permutationsäquivalenzen der Inzidenzmatrix erhalten. Daher interessiert und natürlich, wie sich einfache Permutationsäquivalenzen auf der doppelgeordneten Inzidenzmatrix, die den Rand und die 0-te Blockzeile erhalten, auf die inneren Blockzeilen und ihren zugehörigen lateinischen Quadraten auswirken, um eventuell weitere Festlegungen, insbesondere bei der ersten Blockzeile, vornehmen zu können.

Eine erste solche Permutationsäquivalenz ist das Permutieren der relativen Zeilen, jede Blockzeile mit einer anderen Permutation. Dies läßt sich als Übergang der Inzidenzmatrix I zur Matrix $\mathrm{diag}(E_{n+1}, E_n, P_1, \ldots, P_{n-1})\ I$ schreiben, wobei E_k jeweils die Einheitsmatrix der Größe k ist und $P_1, \ldots, P_{n-1} \in \mathcal{S}_n$ sind. Für die zugehörigen lateinischen Quadrate $A^{(1)}, \ldots, A^{(n-1)}$ bedeutet dies ein Permutieren der Zeilen, also ein Übergang von $A^{(k)}$ zu $P_k A^{(k)}$ für $k = 1, \ldots, n-1$. Zum Zweiten lassen sich auch die Blockspalten permutieren. Dies läßt sich als Übergang der Inzidenzmatrix I zur Matrix $\mathrm{diag}(E_1, P, E_n, \ldots, E_n)\, I\, \mathrm{diag}(E_{n+1}, \mathcal{P})^\top$ schreiben, wobei $P \in \mathcal{S}_n$ und $\mathcal{P} = P \otimes E_n$ sind. Dabei ist für eine $(k \times k)$-Matrix A und eine $(k' \times k')$-Matrix B das **Kronecker-Produkt** $A \otimes B$ durch die $(kk' \times kk')$-Matrix

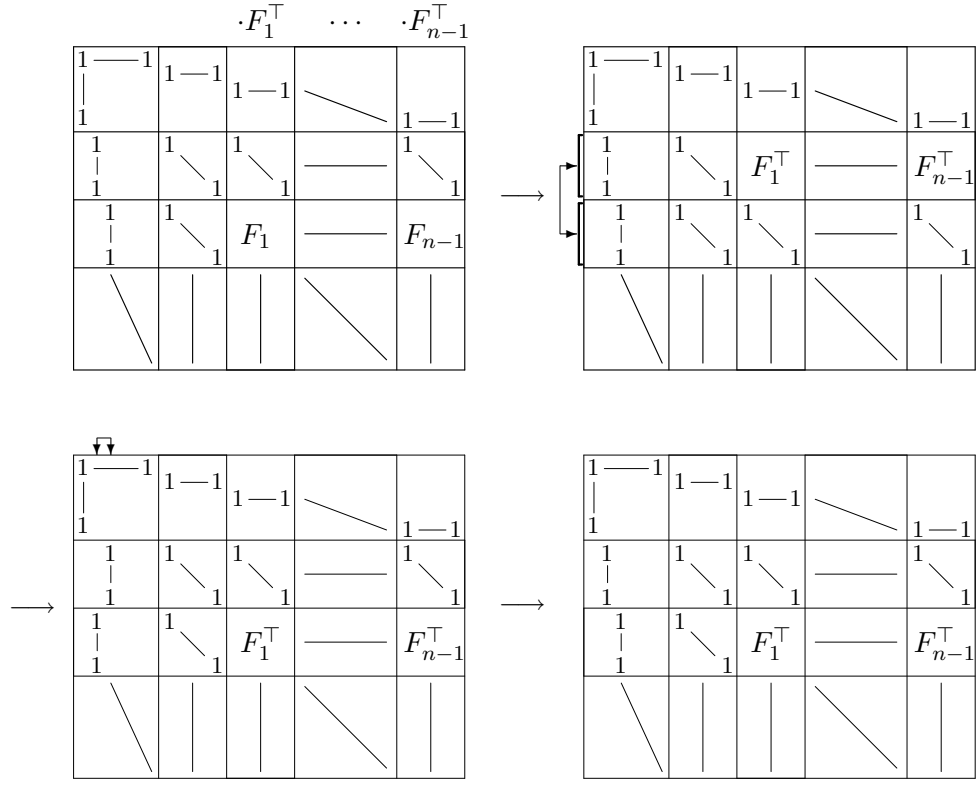
$$A \otimes B := \begin{pmatrix} a_{1,1}B & \cdots & a_{1,l}B \\ \vdots & \ddots & \vdots \\ a_{k,1}B & \cdots & a_{k,l}B \end{pmatrix}$$

definiert (vgl. Hall [9]). Man kann $\mathcal{P}$ auch so ausdrücken, daß man die Einsen der Permutationsmatrix $P \in \mathcal{S}_n$ durch die Einheitsmatrix E_n der Größe n ersetzt und die Nullen durch die Nullmatrix der Größe n. Diese Umformung der Inzidenzmatrix bewirkt für die zugehörigen lateinischen Quadrate $A^{(1)}, \ldots, A^{(n-1)}$ gerade ein Vertauschen der Spalten, also einen Übergang von $A^{(k)}$ zu $A^{(k)}\, P^\top$, wobei P gerade die obige Permutationsmatrix ist. Als dritte Operation betrachten wir die simultane Permutation der relativen Spalten, in jeder Blockspalte mit derselben Permutation P. Das führt zu folgender Transformation:



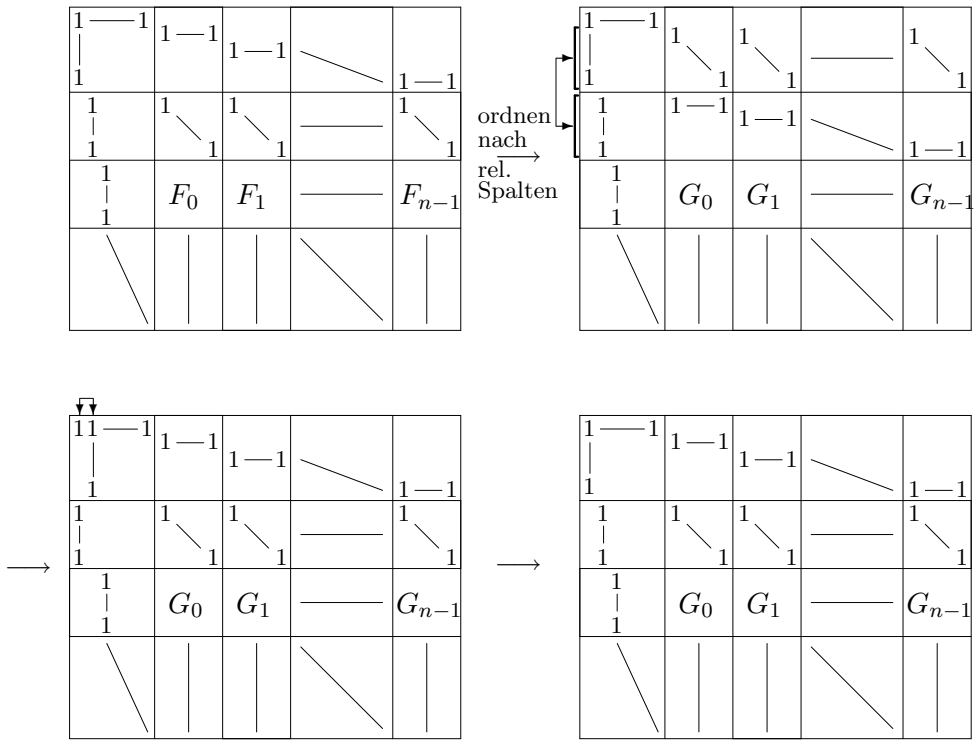
Hierbei ist $P'_{i,j} = P_{i,j} \cdot P^\top = P_{i,j} \cdot P^{-1}$ für $i, j = 1, \ldots, m = n-1$. Diese Permutationsäquivalenz läßt sich als Übergang der Inzidenzmatrix I zur Matrix $\operatorname{diag}(E_{n+1}, P, E_n, \ldots, E_n)\, I \operatorname{diag}(E_{n+1}, P, \ldots, P)^\top$ schreiben. Die zugehörigen lateinischen Quadrate $A^{(1)}, \ldots, A^{(n-1)}$ werden dabei umnummeriert, also $A^{(k)} = (a^{(k)}_{i,j})$ geht über in $(p(a^{(k)}_{i,j}))$, wobei p die zu P gehörende Permutation, definiert durch $p(i) = j \Leftrightarrow P\, e_i = e_j$, ist.

Betrachten wir nur das durch die erste Blockzeile definierte lateinische Quadrat, so können wir mit diesen permutationsäquivalenten Umformungen die Zeilen, die Spalten und die Nummerierung jeweils permutieren. Daher können wir uns bei der Vorgabe des ersten lateinischen Quadrates auf einen Repräsentanten pro Familie beschränken. Weitere Einschränkungen ergeben sich durch folgende Permutationsäquivalenzen:



Die Umformung der Inzidenzmatrix I geschieht dabei durch Übergang auf $\operatorname{diag}(E_{n+1}, W_1, E_n, \ldots, E_n)\, I\, \operatorname{diag}(E_1, W_2, E_{n-2}, E_n, F_1, \ldots, F_{n-1})^\top$, wobei $W_1 = \begin{pmatrix} & E_n \\ E_n & \end{pmatrix} \in \mathcal{S}_{2n}$ und $W_2 = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix} \in \mathcal{S}_2$ sind. Bei dieser Umformung werden die Permutationsmatrizen der ersten Blockzeile transponiert, also die relativen Zeilen mit den relativen Spalten vertauscht. Formuliert auf die lateinischen Quadrate bedeutet dies ein Vertauschen von Zeilen und Nummerierung. Nun gilt bei Permutationsmatrizen $F_i^\top = F_i^{-1}$, daher ist das Vertauschen von Zeilen und Nummerierung bei den lateinischen Quadraten gerade das Invertieren der zu den Spalten des Quadrates gehörenden Permutationen.

Als nächstes wollen wir den Kern nach der relativen Zeile ordnen, d. h. in die i-te Blockspalte kommen alle Vektoren, deren relative Spalte $i+1$ ist (Indexverschiebung da Nummerierung der Blockspalten mit 0 beginnt). Dabei kommen wir zu folgender Umformung:



Diese Umformung können wir als Übergang von der Inzidenzmatrix I zu $\operatorname{diag}(E_1, W_1, E_n, \ldots, E_n)\ I\ \operatorname{diag}(W_2, E_{n-1}, W_3)^\top$ schreiben, wobei die Matrizen W_1 und W_2 die aus der vorigen Umformung sind, und $W_3 \in \mathcal{S}_{n^2}$ definiert ist durch $(W_3)_{i,j} = \begin{cases} 1 & \text{falls } (i,j) = (na+b+1, nb+a+1) \\ & \text{mit geeigneten } a, b \in \{0, \ldots, n-1\} \\ 0 & \text{sonst.} \end{cases}$

Weiterhin ist bei dieser Umformung $F_0 = E$ und G_i entsteht aus $F_0, \ldots, F_{n-1}$,

indem in der j-ten Spalte von G_i der i-te Spaltenvektor von F_j steht. Es werden also die relative Spalte und die Blockspalte vertauscht. Für das zugehörige lateinische Quadrat bedeutet dies gerade das Vertauschen von Nummerierung und Spalten. Zusammen mit dem vorigen Ergebnis können wir also die Zeilen, Spalten und Nummerierung des zu der ersten Blockzeile gehörenden lateinischen Quadrates vertauschen. Daher können wir uns bei der Vorgabe des ersten lateinischen Quadrates sogar auf einen Repräsentanten aus jeder Isomorphieklasse beschränken. Dies teilt unseren Algorithmus zur Berechnung aller Inzidenzmatrizen der Ordnung n in zwei Teile:

i) Berechne die Isomorphieklassen lateinischer Quadrate.

ii) Setze aus jeder Isomorphieklasse ein lateinisches Quadrat als erste Blockzeile ein und berechne den Rest nach Algorithmus 2.16.

4.4 Das Programm-Paket Inzidenz

Eine erste Verwirklichung dieser Algorithmus-Idee findet sich im Programm-Paket Inzidenz. Zum Verständnis seiner Wirkungsweise benötigen wir noch einige speziellere mathematische Vorbereitungen.

Definition 4.6
Sei A eine $n \times m$-Matrix, $m \leq n$, über den Zahlen $1, \ldots, n$, wobei jede Zahl in jeder Zeile und jeder Spalte höchstens einmal vorkommt. Dann heißt A ein **lateinisches Rechteck** der **Ordnung** n und **Länge** m. Sind $f_1, \ldots, f_m$ überdeckungsfreie Permutationen, so bildet die Matrix

$$\tilde{A} := \begin{pmatrix} f_1(1) & \cdots & f_m(1) \\ \vdots & \ddots & \vdots \\ f_1(n) & \cdots & f_m(n) \end{pmatrix}$$

ein lateinisches Rechteck. Umgekehrt lassen sich die Spalten eines lateinischen Rechtecks nach dieser Vorschrift zu m überdeckungsfreien Permutationen transformieren. In diesem Fall identifizieren wir das Tupel $(f_1, \ldots, f_m)$ mit dem lateinischen Rechteck $\tilde{A}$. Weiterhin bezeichnen wir mit $\mathbb{L}_{n,m}$ die Menge aller lateinischer Rechtecke der Ordnung n und Länge m.

Als Spezialfall lassen sich für $m = n$ die lateinischen Quadrate in die lateinischen Rechtecke einreihen (vgl. Hall [9]). Insbesondere ist $\mathbb{L}_n = \mathbb{L}_{n,n}$. Nimmt man nun von einem lateinischen Rechteck die ersten $\tilde{m}$ Spalten, so bilden diese wiederum ein lateinisches Rechteck, ein **Teilrechteck** des ursprünglichen Rechtecks. Daher läßt sich die Gesamtheit der lateinischen Quadrate der Ordnung n berechnen, indem wir einfach, angefangen beim lateinischen Rechteck der Länge 0, immer alle Spalten berechnen, mit der das Rechteck erweitert werden kann, so daß wir ein lateinisches Rechteck mit einer um eins größeren Länge erhalten. Mit allen so erhaltenen Rechtecken wiederholen wir das Verfahren, bis wir Länge n und damit die lateinischen

Quadrate erreicht haben. Dies gibt natürlich für höhere Ordnungen eine nur noch schwer behandelbare Anzahl von lateinischen Quadraten. Wie wir jedoch gezeigt haben, können wir uns bei der Vorgabe der ersten Blockzeile auf jeweils einen Repräsentanten aus jeder Isomorphieklasse von lateinischen Quadraten beschränken. Effektiver wäre es natürlich, schon die lateinischen Rechtecke zu Klassen zusammenzufassen. Dies motiviert folgende Übertragung der Isomorphie auf die lateinischen Rechtecke:

Definition 4.7
Zwei lateinische Rechtecke A und B heißen **isomorph**, wenn jedes lateinische Quadrat, welches A als Teilrechteck enthält, in derselben Isomorphieklasse ist wie ein lateinisches Quadrat, welches B als Teilrechteck enthält, und umgekehrt jedes lateinische Quadrat, welches B als Teilrechteck enthält, in derselben Isomorphieklasse ist, wie ein lateinisches Quadrat, welches A als Teilrechteck enthält.

Die isomorphen lateinischen Rechtecke sind also gerade diejenigen, deren Erweiterungen zu lateinischen Quadraten zu denselben Isomorphieklassen von lateinischen Quadraten führen. Daher können wir uns bei unserem Algorithmus zur Berechnung der Gesamtheit von lateinischen Quadraten (bis auf Isomorphie) auf jeweils einen Repräsentanten pro Isomorphieklasse von Rechtecken beschränken. Für die praktische Zusammenfassung der Rechtecke in die Isomorphieklassen ist die Überprüfung der Bedingungen in der Definition allerdings recht langwierig. Für einen effektiven Algorithmus benötigen wir daher ein Kriterium, mit dem wir die Isomorphie schon an den lateinischen Rechtecken selbst und nicht erst an den lateinischen Quadraten, welche dieses Rechteck als Teilrechteck enthalten, erkennen können. Ein solches Kriterium stellt das folgende Lemma bereit:

Lemma 4.8
Ein lateinisches Rechteck bleibt unter den Operationen Permutieren von Zeilen, Spalten oder Nummerierung, sowie unter Vertauschen von Zeilen und Nummerierung, in seiner Isomorphieklasse, d. h. das lateinische Rechteck $(f_1, \ldots, f_m)$ *der Ordnung* n *ist isomorph zu folgenden lateinischen Rechtecken:*

i) $(p\,f_1, \ldots, p\,f_m)$ *mit* $p \in S_n$
ii) $(f_{q(1)}, \ldots, f_{q(m)})$ *mit* $q \in S_m$
iii) $(f_1\,p, \ldots, f_m\,p)$ *mit* $p \in S_n$
iv) $(f_1^{-1}, \ldots, f_m^{-1})$

Insbesondere ist das Rechteck $(f_1, \ldots, f_m)$ *isomorph zum konjugierten Rechteck* $(p\,f_1\,p^{-1}, \ldots, p\,f_m\,p^{-1})$ *mit* $p \in S_n$.

Bevor wir dieses Lemma beweisen, untersuchen wir, ähnlich wie bei den lateinischen Quadraten, eine Gruppe, welche von diesen Operationen erzeugt wird und auf $\mathbb{L}_{n,m}$ wirkt. Sei hierzu U_1' die Gruppe, welche aus den

Permutationen der Zeilen, U_2' die Gruppe, welche aus den Permutationen der Spalten und U_3' die Gruppe, welche aus den Umnummerierungen besteht. Dann ist die Wirkung von $U' := U_1' \times U_2' \times U_3' \cong S_n \times S_m \times S_n$ auf $\mathbb{L}_{n,m}$ wohldefiniert. Besteht noch $V' \cong S_2$ aus der Vertauschung von Zeilen und Nummerierung, so ist die Wirkung des semidirekten Produkts $T_m' := U' \underset{\varphi}{\rtimes} V'$ auf $\mathbb{L}_{n,m}$ wohldefiniert. Sind ϑ_i die Isomorphismen von S_n bzw. S_m in $U_i' \subseteq U'$und fassen wir $p \in V'$ als Permutation der Zahlen $\{1,3\}$ auf, so ist hierbei der Homomorphismus φ für $\alpha, \gamma \in S_n$ und $\beta \in S_m$ definiert durch $\varphi(p)(\vartheta_1(\alpha)\vartheta_2(\beta)\vartheta_3(\gamma)) = \vartheta_{p(1)}(\alpha)\vartheta_2(\beta)\vartheta_{p(3)}(\gamma)$. Damit läßt sich der Beweis wie folgt formulieren:

Beweis:
Wir müssen zeigen, daß die Bahnen von T_m' alle jeweils in einer Isomorphieklasse von $L_{n,m}$ verlaufen. Dies folgt jedoch einfach daraus, daß wir T_m' kanonisch in T einbetten können. Denn sind zwei lateinische Rechtecke in derselben Bahn von T_m', so führt jede Erweiterung des einen Rechtecks zu einem lateinischen Quadrat unter der Wirkung von $T_m' \leq T$ zu einem isomorphen Quadrat, welches das andere Rechteck als Teilrechteck enthält und umgekehrt. Die Rechtecke sind somit isomorph.

□

Bei unserem Algorithmus wollen wir ein lateinisches Rechteck $(f_1, \ldots, f_{m-1})$ der Länge $m-1$ auf ein Rechteck $(f_1, \ldots, f_{m-1}, f_m)$ der Länge m erweitern. Hierbei wollen wir Permutationen, welche, in der m-ten Spalte eingesetzt, zu derselben Isomorphieklasse von Rechtecken führen, in einer Klasse zusammenfassen, von welcher wir bei erneuter Erweiterung des Rechtecks nur eine Permutation betrachten müssen. Das Finden einer solchen Klasse (oder genauer einer Unterklasse der ganzen Klasse) läßt sich durch Betrachten der Bahnen der Wirkung einer geeigneten Untergruppe von T_m' auf $L_{n,m}$, welche das Teilrechteck $(f_1, \ldots, f_{m-1})$ festläßt, erreichen. Für einen Algorithmus geeigneter ist es, wenn diese Untergruppe nicht von der einzusetzenden Permutation f_m abhängt, also sogar Untergruppe von T_{m-1}' ist. Man beachte hierbei, daß der Unterschied zwischen T_m' und T_{m-1}' nur darin besteht, daß im einen Fall die Permutation q, welche die Spalten permutiert, aus S_m und im anderen Fall aus S_{m-1} ist. Die Wirkung von geeigneten Untergruppen von T_{m-1}' führen zu folgenden Lemmata:

Lemma 4.9
Die beiden lateinischen Rechtecke $(f_1, \ldots, f_m)$ *und* $(f_1, \ldots, f_{m-1}, z\, f_m\, z^{-1})$, *mit einem* z *aus dem Normalisator* $N_{S_n}(f_1, \ldots, f_{m-1})$, *sind isomorph.*
Insbesondere sind $(f_1, \ldots, f_{m-1}, f_m)$ *und* $(f_1, \ldots, f_{m-1}, z\, f_m\, z^{-1})$ *für* z *aus dem Zentralisator* $C_{S_n}(f_1, \ldots, f_{m-1})$ *isomorph.*

Beweis:

Es wirkt $z \in N_{S_n}(f_1, \ldots, f_{m-1})$ per Konjugation auf den Permutationen $\{f_1, \ldots, f_{m-1}\}$. Daher ist $(z\, f_1\, z^{-1}, \ldots, z\, f_{m-1}\, z^{-1}) = (f_{q(1)}, \ldots, f_{q(m-1)})$ mit einem geeigneten $q \in S_{m-1}$. Das lateinische Rechteck $(f_1, \ldots, f_m)$ ist deshalb isomorph zu dem Rechteck $(z\, f_1\, z^{-1}, \ldots, z\, f_{m-1}\, z^{-1}, z\, f_m\, z^{-1}) = (f_{q(1)}, \ldots, f_{q(m-1)}, z\, f_m\, z^{-1})$ und dies ist isomorph zu $(f_1, \ldots, f_{m-1}, z\, f_m\, z^{-1})$ nach Lemma 4.8. Die zweite Aussage folgt unmittelbar daraus, da der Normalisator den Zentralisator enthält.

□

Die Aussagen über den Zentralisator ist an dieser Stelle aus technischen Gründen hinzugenommen, da sich $C_{S_n}(f_1, \ldots, f_{m-1})$ aus $C_{S_n}(f_1, \ldots, f_{m-2})$ recht einfach berechnen läßt, indem man die Elemente von $C_{S_n}(f_1, \ldots, f_{m-2})$ überprüft, ob sie mit f_{m-1} kommutieren, wogegen die Berechnung des Normalisators aufwendiger ist. Dies wird jedoch noch an späterer Stelle näher beleuchtet und wir betrachten nun noch einige spezielle Isomorphien:

Lemma 4.10
Alle lateinischen Rechtecke der Länge 1 *sind isomorph zu* (id).
Haben f *und* g *dieselbe Zyklenstruktur, so sind* (id, f) *und* (id, g) *isomorph.*

Beweis:

Ein lateinisches Rechteck (f) der Länge 1 ist nach Lemma 4.8 isomorph zu $(f^{-1}\, f) = (\mathrm{id})$. Haben f und g dieselbe Zyklenstruktur, so gibt es ein $z \in S_n$, so daß $z\, f\, z^{-1} = g$, und nach Lemma 4.9 ist dann (id, f) isomorph zu $(\mathrm{id}, z\, f\, z^{-1}) = (\mathrm{id}, g)$.

□

Ein weiterer Spezialfall ergibt sich für Rechtecke der Länge 3:

Lemma 4.11
Sei (f_1, f_2, f_3) *ein lateinisches Rechteck mit* $f_1 = \mathrm{id}$ *und* $t \in S_n$ *eine Permutation mit* $t\, f_2\, t^{-1} = f_2^{-1}$. *Dann sind folgende lateinische Rechtecke isomorph:*

i) (f_1, f_2, f_3)
ii) $(f_1, f_2, t\, f_3^{-1}\, t^{-1})$
iii) $(f_1, f_2, f_2\, f_3^{-1})$
iv) $(f_1, f_2, t\, f_3\, f_2^{-1}\, t^{-1})$

Beweis:

Sei (f_1, f_2, f_3) ein lateinisches Rechteck mit $f_1 = \mathrm{id}$ und $t \in S_n$ eine Permutation mit $t\, f_2^{-1}\, t^{-1} = f_2$. Eine solche Permutation existiert immer, da f_2 und f_2^{-1} dieselbe Zyklenstruktur haben und daher konjugiert zueinander sind. Dann ist (f_1, f_2, f_3) isomorph zu $(f_1^{-1}, f_2^{-1}, f_3^{-1}) = (f_1, f_2^{-1}, f_3^{-1})$

nach Lemma 4.8 iv). Konjugieren wir die Permutationen mit t, so erhalten wir nach Lemma 4.8 die Isomorphie mit $(t\,f_1\,t^{-1}, t\,f_2^{-1}\,t^{-1}, t\,f_3\,t^{-1}) = (f_1, f_2, t\,f_3^{-1}\,t^{-1})$, womit ii) gezeigt ist. Multiplizieren wir dagegen das lateinische Rechteck $(f_1, f_2^{-1}, f_3^{-1})$ mit f_2 von links, so erhalten wir nach Lemma 4.8 i) die Isomorphie mit $(f_2\,f_1, f_2\,f_2^{-1}, f_2\,f_3^{-1}) = (f_2, f_1, f_2\,f_3^{-1})$. Dieses Rechteck wiederum ist nach Lemma 4.8 ii) isomorph zu $(f_1, f_2, f_2\,f_3^{-1})$, womit iii) gezeigt ist. Die Umformung iv) folgt aus der Hintereinanderausführung von iii) und ii).

□

In den letzten Lemmata haben wir nun eine Reihe von isomorphen Umformungen kennengelernt, welche es uns erlauben, die lateinischen Rechtecke in Klassen isomorpher Rechtecke zusammenzufassen. Insbesondere haben wir für ein lateinisches Rechteck (f_1, f_2, f_3) der Länge 3 mit $f_1 = \mathrm{id}$ eine unüberschaubare Anzahl von Kombinationsmöglichkeiten von diesen Umformungen. Für einen effektiven Algorithmus müssen wir uns dabei überlegen, welche Kombinationen dieser Umformungen wirklich nötig sind, um die gesamte Klasse zu erhalten. Hierbei ist es aufgrund der geringen Länge von besonderer Bedeutung, daß wirklich die gesamte Klasse zusammengefaßt wird, da schon eine geringe Erhöhung der lateinischen Rechtecke von kleiner Länge die Anzahl der erhaltenen lateinischen Quadrate stark erhöht. Damit beschäftigt sich das folgende Lemma:

Lemma 4.12
Sei (f_1, f_2, f_3) ein lateinisches Rechteck mit $f_1 = \mathrm{id}$ und sei $K := \{\tilde{f}_3 \mid (f_1, f_2, \tilde{f}_3)$ ist in derselben Bahn wie (f_1, f_2, f_3) unter der Wirkung von T_2' auf $L_{n,3}\}$. Dann ist $K = \{z\,f\,z^{-1} \mid z \in C_{S_n}(f_2)$ und $f \in \{f_3, t\,f_3^{-1}\,t^{-1}, f_2\,f_3^{-1}, t\,f_3\,f_2^{-1}\,t^{-1}\}\}$, wobei t eine Permutation ist mit $t\,f_2^{-1}\,t^{-1} = f_2$.

Beweis:

Sei K wie im Lemma definiert und $\tilde{K} := \{zfz^{-1} \mid z \in C_{S_n}(f_2)$ und $f \in \{f_3, t\,f_3^{-1}\,t^{-1}, f_2\,f_3^{-1}, t\,f_3\,f_2^{-1}\,t^{-1}\}\}$, wobei t eine beliebige Permutation ist mit $t\,f_2^{-1}\,t^{-1} = f_2$. Weiterhin identifizieren wir T_2' mit ihrer Einbettung in T_3'. Die Inklusion $\tilde{K} \subseteq K$ folgt dann einfach daraus, daß die Umformungen in Lemma 4.9 und Lemma 4.11 für ein Rechteck der Länge 3 in T_2' enthalten sind und Kombinationen dieser Umformungen die verwendeten Umformungen von (f_1, f_2, f_3) zu (f_1, f_2, f) mit $f \in \tilde{K}$ bewirken.

Für die Umkehrung untersuchen wir zunächst, inwieweit $\tilde{K}$ von der speziellen Wahl von t abhängt. Hierzu sei $t_1\,f_2^{-1}\,t_1^{-1} = f_2 = t_2\,f_2^{-1}\,t_2^{-1}$. Dann ist $(t_1\,t_2^{-1})\,f_2\,(t_1\,t_2^{-1})^{-1} = t_1\,(t_2^{-1}\,f_2\,t_2)\,t_1^{-1} = t_1\,f_2^{-1}\,t_1^{-1} = f_2$, also $t_1\,t_2^{-1} \in C_{S_n}(f_2)$ oder $t_1 = z\,t_2$ mit geeignetem $z \in C_{S_n}(f_2)$. Dies zeigt, daß $\tilde{K}$ durch das Konjugieren mit dem Zentralisator $C_{S_n}(f_2)$ unabhängig von der speziellen Wahl von t ist, wir benötigen dieses Ergebnis jedoch auch im folgenden Nachweis der umgekehrten Inklusion. Sei hierzu $B :=$

$\{(p_1\, f_{q(1)}\, p_2^{-1}, p_1\, f_{q(2)}\, p_2^{-1}, p_1\, f_3\, p_2^{-1}),\ \ (p_1\, f_{q(1)}^{-1}\, p_2^{-1}, p_1\, f_{q(2)}^{-1}\, p_2^{-1}, p_1\, f_3^{-1}\, p_2^{-1}) \mid p_1, p_2 \in S_n, q \in S_2\}$ die Bahn des Rechtecks (f_1, f_2, f_3) unter der Wirkung von T_2'. Dann ist die Menge $K = \{f \mid (f_1, f_2, f) \in B\}$. Betrachten wir hierzu die einzelnen Elemente von B, wobei wir nach $q \in S_2$ unterscheiden.

1. Fall: $q = \mathrm{id}$:

- $(p_1\, f_1\, p_2^{-1}, p_1\, f_2\, p_2^{-1}, p_1\, f_3\, p_2^{-1}) = (f_1, f_2, f)$:
  Aufgrund der ersten Spalte $p_1\, f_1\, p_2^{-1} = p_1\, p_2^{-1} = \mathrm{id} = f_1$ muß $p_1 = p_2$ gelten. Aus der zweiten Spalte $p_1\, f_2\, p_1^{-1} = f_2$ folgt daraus $p_1 \in C_{S_n}(f_2)$. Damit ist $f = z\, f_3\, z^{-1}$ mit geeignetem $z \in C_{S_n}(f_2)$ und damit $f \in \tilde{K}$.
- $(p_1\, f_1^{-1}\, p_2^{-1}, p_1\, f_2^{-1}\, p_2^{-1}, P_1\, f_3^{-1}\, p_2^{-1}) = (f_1,\ f_2,\ f)$:
  Wegen der ersten Spalte ist wiederum $p_1 = p_2$ und daher $p_1\, f_2^{-1}\, p_1^{-1} = f_2$, also $p_1 = z\, t$ mit geeignetem $z \in C_{S_n}(f_2)$. Dann ist jedoch $f = (zt)\, f_3^{-1}\, (zt)^{-1} = z(t\, f_3\, t^{-1})\, z^{-1} \in \tilde{K}$.

2. Fall: $q = (1\,2)$:

- $(p_1\, f_2\, p_2^{-1}, p_1\, f_1\, p_2^{-1}, p_1\, f_3\, p_2^{-1}) = (f_1, f_2, f)$:
  Die erste Spalte $p_1\, f_2\, p_2^{-1} = (p_1\, f_2) p_2^{-1} = f_1 = \mathrm{id}$ impliziert $p_2 = p_1\, f_2$. Wegen der zweiten Spalte $p_1\, f_1\, (p_1\, f_2)^{-1} = p_1\, f_2^{-1}\, p_1^{-1} = f_2$ ist weiterhin $p_1 = z\, t$ mit geeignetem $z \in C_{S_n}(f_2)$. Dann ist jedoch $f = (z\, t)\, f_3 (z\, t\, f_2)^{-1} = z\, (t\, f_3\, f_2^{-1}\, t^{-1})\, z^{-1} \in \tilde{K}$.
- $(p_1\, f_2^{-1}\, p_2^{-1}, p_1\, f_1^{-1}\, p_2^{-1}, p_1\, f_3^{-1}\, p_2^{-1}) = (f_1, f_2, f)$:
  Aufgrund der zweiten Spalte $p_1\, f_1^{-1}\, p_2^{-1} = p_1\, p_2^{-1} = f_2$ muß $p_1 = f_2\, p_2$ gelten. Aus der ersten Spalte $f_2\, p_2\, f_2^{-1}\, p_2^{-1} = f_1 = \mathrm{id}$ folgt dann $p_2 \in C_{S_n}(f_2)$. Für ein geeignetes $z \in C_{S_n}(f_2)$ ist daher $f = (f_2\, z)\, f_3^{-1}\, z^{-1} = z\, (f_2\, f_3^{-1})\, z^{-1} \in \tilde{K}$.

Somit ist auch die Inklusion $K \subseteq \tilde{K}$ gezeigt.

□

Kommen wir nun zu einer anderen Idee, nämlich eine Ordnung auf den lateinischen Rechtecken zu definieren. Dies können wir nutzen, indem wir immer jeweils das kleinste Rechteck einer Isomorphieklasse mit weiteren Spalten erweitern. Stellen wir dann fest, daß sich ein lateinisches Rechteck durch isomorphe Umformungen in ein kleineres Rechteck umwandeln läßt, so ist das ursprüngliche Rechteck nicht das Kleinste der Isomorphieklasse, ist also schon behandelt worden, und wir können es von weiteren Berechnungen ausschließen. Ein wesentlicher Teil der besprochenen Umformungen basiert jedoch auf Konjugation der durch die Spalten beschriebenen Permutationen mit geeigneten Permutationen aus S_n. Da hierbei die Zyklenstrukturen erhalten bleiben, empfiehlt es sich, für die lateinischen Rechtecke eine Ordnung zu wählen, welche die Zyklenstruktur mitberücksichtigt. Dies führt zu folgender Definition:

Definition 4.13
Für eine Permutation f sei die Zyklenstruktur **Zykl(f)** der Vektor der Zyklenlängen von f, geordnet nach der Größe. Sei weiterhin eine Ordnung auf den verschiedenen Zyklenstrukturen der Permutationen von S_n gegeben. Dann definieren wir auf den Permutationen von S_n eine Ordnungsrelation, die **Zyklenstrukturordnung** oder kurz **Ordnung** durch
$f < g :\Leftrightarrow (\text{Zykl}(f), f(1), \ldots, f(n)) < (\text{Zykl}(g), g(1), \ldots, g(n))$ nach der lexikographischen Ordnung.
Mit dieser Ordnung definieren wir eine **Ordnung** für lateinische Rechtecke durch
$(f_1, \ldots, f_m) < (g_1, \ldots, g_m) :\Leftrightarrow (f_1, \ldots, f_m) < (g_1, \ldots, g_m)$ nach der lexikographischen Ordnung. Hierbei betrachten wir $(f_1, \ldots, f_m)$ und $(g_1, \ldots, g_m)$ auf der linken Seite als lateinische Rechtecke und auf der rechten Seite als die zugehörigen Tupel von Permutationen.

Man beachte hierbei, daß diese Ordnung auf den Permutationen bzw. lateinischen Rechtecken natürlich von der vorher gegebenen Ordnung der Zyklenstrukturen abhängt. Die offensichtlichste Ordnung ist dabei die lexikographische Ordnung der zugehörigen Vektoren der Zyklenlängen, doch werden wir noch sehen, daß es für unser Problem geeignetere Ordnungen auf den Zyklenstrukturen gibt.

Kommen wir nun zu einem anderen Problem, welches wir schon angesprochen haben, nämlich der Berechnung des Normalisators. Eine frühe Version des Programms berechnete den Normalisator, indem jede Permutation von S_n überprüft wurde, ob sie normal bezüglich der Permutationen des gegebenen lateinischen Rechtecks ist. Dies sind für höhere Ordnungen recht viele zu überprüfende Permutationen, wodurch die Rechenzeit erheblich erhöht wird. Um diese zu untersuchenden Permutationen stärker einzuschränken, benutzen wir die folgenden beiden einfachen Lemmata.

Lemma 4.14
Seien $f, g, h \in S_n$ Permutationen, seien s eine Permutation mit $s\,f\,s^{-1} = h$ und t eine Permutation mit $t\,h\,t^{-1} = g$. Dann ist $\{u \in S_n \mid u\,f\,u^{-1} = g\} = \{t\,z\,s \mid z \in C_{S_n}(h)\}$. Insbesondere ist $C_{S_n}(g) = \{t\,z\,t^{-1} \mid z \in C_{S_n}(h)\}$.

Beweis:

Für $f, g, h \in S_n$ seien $s\,f\,s^{-1} = h$ und $t\,h\,t^{-1} = g$. Für $z \in C_{S_n}(h)$ ist dann $(t\,z\,s)\,f\,(t\,z\,s)^{-1} = t\,z\,s\,f\,s^{-1}\,z^{-1}\,t^{-1} = t\,z\,h\,z^{-1}\,t^{-1} = t\,h\,t^{-1} = g$. Ist umgekehrt $u\,f\,u^{-1} = g$, so ist $(t^{-1}\,u\,s^{-1})\,h\,(t^{-1}\,u\,s^{-1})^{-1} = t^{-1}\,u\,s^{-1}\,h\,s\,u^{-1}\,t = t^{-1}\,u\,f\,u^{-1}\,t = t^{-1}\,g\,t = h$, also $u = t\,z\,s$ mit geeignetem $z \in C_{S_n}(h)$. Die zweite Aussage erhalten wir daraus mit $g = f$.

□

Lemma 4.15
Seien $f_1, \ldots, f_m \in S_n$. Sei $F := \{f_i \mid i = 1, \ldots, m-1, \text{Zykl}(f_i) = \text{Zykl}(f_m)\}$ die Menge der Permutationen f_i, $i < m$ mit derselben Zyklenstruktur wie f_m. Sei h eine weitere Permutation mit derselben Zyklenstruktur wie f_m, sei t eine Permutation mit $t\, f_m\, t^{-1} = h$ und seien s_i Permutationen mit $s_i\, h\, s_i^{-1} = f_i$ für $f_i \in F$. Dann gilt für den Normalisator:

$$N_{S_n}(f_1, \ldots, f_m) \subseteq (N_{S_n}(f_1, \ldots, f_{m-1}) \cap C_{S_n}(f_m)) \cup \bigcup_{f_i \in F} \{t\, z\, s_i \mid z \in C_{S_n}(h)\}$$

Beweis:

Der Normalisator $N := N_{S_n}(f_1, \ldots, f_m)$ wirkt per Konjugation auf der Menge $\{f_1, \ldots, f_m\}$. Bildet dabei $u \in N$ per Konjugation f_m auf sich ab, also $u\, f_m\, u^{-1} = f_m$, so ist $u \in C_{S_n}(f_m)$. Andererseits permutiert u per Konjugation die Permutationen $\{f_1, \ldots, f_{m-1}\}$ untereinander, also $u \in N_{S_n}(f_1, \ldots, f_{m-1})$. Insgesamt ist somit $u \in N_{S_n}(f_1, \ldots, f_{m-1}) \cap C_{S_n}(f_m)$. Bildet dagegen u per Konjugation f_m auf ein f_i, $i < m$ ab, so muß f_i dieselbe Zyklenstruktur wie f_m haben, also $f_i \in F$. Nach Lemma 4.14 ist damit $u \in \{t\, z\, s_i \mid z \in C_{S_n}(h)\}$, t, s_i entsprechend dem Lemma, da dies alle Permutationen enthält, welche f_m per Konjugation auf f_i abbilden.

□

Mit diesem Lemma können wir unseren Algorithmus zur Berechnung des Normalisators $N_{S_n}(f_1, \ldots, f_m)$ viel effektiver gestalten, da wir nicht mehr alle $n!$ Permutationen von S_n überprüfen müssen, ob sie normal bezüglich $f_1, \ldots, f_m$ sind, sondern nur diese wesentlich kleinere Menge in dem Lemma. Hierbei können wir diese Menge mit akzeptablen Aufwand bestimmen, denn der Normalisator $N_{S_n}(f_1, \ldots, f_{m-1})$ wird sowieso schon vorher berechnet und wir müssen dessen Elemente nur noch überprüfen, ob sie mit f_m kommutieren. Für die Mengen $\{t\, z\, s_i \mid z \in C_{S_n}(h)\}$ benötigen wir den Zentralisator einer beliebigen Permutation h, welche dieselbe Zyklenstruktur wie f_m hat. Hierzu legen wir am Anfang zu jeder Zyklenstruktur eine standardisierte Permutation h_j mit dieser Zyklenstruktur fest und berechnen deren Zentralisator $C_{S_n}(h_j)$. Dann müssen wir zu f_m nur dasjenige h_j mit derselben Zyklenstruktur suchen, ein $t \in S_n$ berechnen mit $t\, f_m\, t^{-1} = h_j$ und zu jedem $f_i \in F$ ein $s_i \in S_n$ mit $s_i\, h_j\, s_i^{-1} = f_i$. Dies läßt sich mit akzeptablen Aufwand durchführen und nach ein paar Multiplikationen haben wir die Kandidaten für den Normalisator wesentlich eingeschränkt, so daß wir den Normalisator in für unser Problem ausreichender Zeit berechnen können.

Nach diesen Vorbereitungen können wir nun mit der Besprechung der einzelnen Programmteile beginnen. Diese sind in folgende Dateien aufgeteilt:

Main.cpp
Das Hauptprogramm. Hier wird das Programm gesteuert und die gesamtgültigen Konstanten vereinbart. Die Ordnung der zu berechnenden projektiven Ebene setzt man durch Einbinden der entsprechenden Datei ordnung3.h,. . . , ordnung9.h fest.

ordnung3.h, . . . , ordnung9.h
Diese Dateien enthalten die Ordnung der zu berechnenden projektiven Ebene, als Konstante DIM, und die verschiedenen Zyklenstrukturen von fixpunktfreien Permutationen dieser Länge. Wird der Kürze halber in Zukunft ordnung.h genannt.

zentral.h
Die Klasse Zentral, welche die lateinischen Quadrate der ersten Blockzeile auf jeweils möglichst wenige Repräsentanten einer Isomorphieklasse einschränkt.

perm.h
Die Klasse Perm, eine Hilfsklasse für zentral.h zur Speicherung einer Permutation mit ihrer Zyklenstruktur.

simpelPerm.h
Die Klasse SimpelPerm, eine Hilfsklasse für zentral.h zur Speicherung einer Permutation ohne ihre Zyklenstruktur.

matrix.h
Die Klasse Matrix zur Berechnung der fertigen Inzidenzmatrizen bei gegebener ersten Blockzeile.

matrixEl.h
Die Klasse MatrixElement, eine Hilfsklasse zu matrix.h zur effektiven Behandlung eines Zeilenabschnitts.

func.h
Eine Sammlung von Hilfsfunktionen, welche zu keiner der Klassen passen.

getclock.h
Funktionen von F. Schmitt zur Auswertung der benötigten Prozessorzeit.

newmath.h
Funktionen von M. Baumann zur Speicherung einer Matrix im pbm-Format.

Kommen wir nun zur praktischen Umsetzung des Algorithmus. Hierbei betrachten wir zunächst den ersten Teil des Algorithmus:

Die Reduzierung der lateinischen Anfangsquadrate

In der Klasse Zentral sind die benötigten Funktionen zur Reduzierung der lateinischen Anfangsquadrate auf möglichst wenige Repräsentanten einer Isomorphieklasse zusammengetragen. Hierfür benötigen wir eine geeigne-

te Beschreibung der Permutationen. Dies geschieht in den folgenden beiden Klassen:

SimpelPerm
Hierin wird eine Permutation $p \in S_n$ in einem Array zahlen gespeichert, so daß $p(i) =$ zahlen[i] für alle Zahlen $i = 0, \ldots, n-1$. Dabei fassen wir p als Permutation der Zahlen $0, \ldots, n-1$ auf, da die Indizierung eines Arrays unter C++ mit 0 beginnt. Diese Klasse benutzen wir zur Beschreibung von Permutationen, für die die Kenntnis der Zyklenstruktur unnötig ist, wie zum Beispiel für die Permutationen des Normalisators.

Perm
Die Permutationen dieser Klasse enthalten neben der Funktionalität, welche die Klasse SimpelPerm anbietet, noch die Speicherung der Zyklenstruktur der Permutation. Dazu werden am Anfang die verschiedenen fixpunktfreien Zyklenstrukturen in der Datei ordnung.h festgelegt. Die Zyklenstrukturen sind dabei durch die geordneten Arrays der Zyklenlängen eindeutig bestimmt, welche wir im statischen zweidimensionalen Array Perm::klammern speichern. Die Zyklenstruktur einer Permutation beschreiben wir dann durch die Objekt-Variable ordnung, welche so zu verstehen ist, daß die Variable Perm::klammern[ordnung] das geordnete Array der Zyklenlängen der Permutation enthält. Hierbei ist die in Definition 4.13 angesprochene Ordnung auf den Zyklenstrukturen, welche auch die Ordnung auf den Rechtecken beeinflußt, dadurch gegeben, daß die Zyklenstruktur derjenigen Permutation kleiner ist, deren Variable ordnung kleiner ist. Somit ist auch diese Ordnung durch die Datei ordnung.h festgelegt. Weiterhin berechnen wir am Anfang zu jeder fixpunktfreien Zyklenstruktur den Zentralisator der kleinsten Permutation dieser Zyklenstruktur und speichern diesen in der statischen Variable normalisator als Array von Listen, um später die effektivere Berechnung des Normalisators mit Hilfe von Lemma 4.15 zu ermöglichen. Die Klasse benutzen wir zur Beschreibung von Permutationen, bei denen die Kenntnis der Zyklenstruktur wichtig ist, also bei allen Permutationen, welche eine Spalte eines lateinischen Rechtecks darstellen sollen.

Kommen wir nun zurück zur praktischen Verwirklichung unseres Algorithmus. Hierbei beginnen wir mit den lateinischen Rechtecken der Länge 1, reduzieren die als isomorph erkannten Rechtecke auf jeweils einen Repräsentanten und fügen an diese Repräsentanten, mit dem kleinsten beginnend, jeweils eine Spalte hinzu. Diese Schritte wiederholen wir, bis wir Rechtecke der Länge n, also die lateinischen Quadrate erreicht haben. Nach Lemma 4.10 können wir uns bei der ersten Spalte auf id und bei der zweiten Spalte auf einen Repräsentanten jeder fixpunktfreien Zyklenstruktur beschränken. Als Repräsentanten wählen wir dabei die jeweils kleinste Permutation der Zyklenstruktur. Mit diesen Vorgaben wenden wir unseren Algorithmus an, indem wir mit der Funktion stepForward() eine Spalte anfügen und mit der Funktion stepBack() die jeweils letzte Spalte des Rechtecks löschen.

stepForward()
Bei der Erweiterung des Rechtecks $(f_1, \ldots, f_{i-1})$ um eine i-te Spalte sind bereits normalisator[i-1], der Normalisator $N_{S_n}(f_1, \ldots, f_{i-2})$, und gueltigPerm[i-1], eine Liste, die alle Permutationen enthält, welche in die $i-1$-te Spalte eingesetzt nicht die Überdeckungsfreiheit verletzen oder zu einer Isomorphieklasse führen, welche schon behandelt wurde. Zunächst berechnen wir normalisator[i], den Normalisator $N_{S_n}(f_1, \ldots, f_{i-1})$. Diese Berechnung führen wir in der Funktion initZentralisator() durch, falls die Zyklenstruktur von f_{i-1} von den Zyklenstrukturen der anderen Permutationen verschieden ist, ansonsten wird der Normalisator in initNormalisator() berechnet. Dann kopieren wir die Permutationen von gueltigPerm[i-1] in die Listen gueltigPerm[i] und tempGueltigPerm. Hierbei speichern wir in diesen Listen nicht die Permutationen selbst, sondern, um Speicherplatz zu sparen, nur Zeiger auf die Permutationen. Die Permutationen selbst sind dabei, geordnet nach unserer Zyklenstrukturordnung, in der Liste permutationen abgelegt. Diese Ordnung überträgt sich auf die Listen gueltigPerm[i] und tempGueltigPerm, wodurch sich die späteren Löschaktionen sehr viel effektiver gestalten lassen. Nun beginnen wir mit der kleinsten Permutation aus tempGueltigPerm und berechnen in der Funktion initKonjKlasse() eine Klasse von Permutationen, welche in die i-te Spalte eingesetzt zu derselben Isomorphieklasse führen. In konjKlasseGueltig() überprüfen wir dann, ob die Klasse Permutationen enthält, welche der Überdeckungsfreiheit widersprechen oder aber zu schon behandelten lateinischen Rechtecken führen. Kommt es zu einem Widerspruch, so löschen wir die Klasse aus gueltigPerm[i] und aus tempPerm, ansonsten löschen wir sie nur aus tempPerm und fügen die kleinste Permutation der Klasse zur Liste repraesSystem[i] hinzu. Dies wiederholen wir mit der nun neuen kleinsten Permutation aus tempGueltigPerm, bis die Liste leer ist. Dann enthält die Liste repraesSystem[i] ein Repräsentantensystem von Permutationen, so daß jedes Rechteck $(f_1, \ldots, f_{i-1}, f)$ zu $(f_1, \ldots, f_{i-1}, f_i)$ mit geeignetem $f_i \in$ repraesSystem[i] isomorph ist, oder aber in der Isomorphieklasse eines schon behandelten Rechtecks liegt. Die kleinste Permutation aus repraesSystem[i] setzen wir dann in die i-te Spalte ein und versuchen erneut, mit dieser Methode eine weitere Spalte anzufügen. Vervollständigen wir hiermit das Rechteck zu einem lateinischen Quadrat, so starten wir in der Funktion konfigAusgabe() den zweiten Teil des Algorithmus, die Vervollständigung der Inzidenzmatrix mit dem Backtracking-Verfahren.

stepBack()
Führt keine Erweiterung des Rechtecks zu einem gültigen Ergebnis, ist also repraesSystem[i] leer, oder ist das Rechteck zu einem lateinischen Quadrat vervollständigt worden, so müssen wir die letzte Spalte des Rechtecks in der Funktion stepBack() löschen. Hierzu entfernen wir die kleinste Permutation aus repraesSystem[i-1], berechnen mit initKonjKlasse() erneut die zugehörige Klasse und löschen diese aus gueltigPerm[i-1]. Die erneute Berechnung

der Klasse ist hierbei angebracht, da wir aus Speicherplatzgründen nicht zu jedem Repräsentanten die ganze Klasse im Speicher behalten können und sich zudem der Aufwand für die erneute Berechnung in Grenzen hält. Ist nun repraesSystem[i-1] leer, so müssen wir weitere Spalten löschen. Ansonsten setzen wir die neue kleinste Permutation von repraesSystem[i-1] in die $(i-1)$-te Spalte ein und versuchen erneut in der Funktion stepForward() das Rechteck um eine Spalte zu erweitern.

initZentralisator()
Hat f_j eine von f_{i-1} verschiedene Zyklenstruktur für alle $j < i-1$, so gilt für den Normalisator $N_{S_n}(f_1, \ldots, f_{i-1}) = N_{S_n}(f_1, \ldots, f_{i-2}) \cap C_{S_n}(f_{i-1})$. Hierbei ist der Normalisator $N_{S_n}(f_1, \ldots, f_{i-2})$ schon durch normalisator[i-1] gegeben, so daß wir deren Permutationen nur noch überprüfen müssen, ob sie mit f_{i-1} kommutieren. Dies ist natürlich weit weniger aufwendig als die folgende Normalisatorberechnung in initNormalisator(), so daß eine gesonderte Behandlung gerechtfertigt ist.

initNormalisator()
Gibt es ein f_j, $j < i-1$, welches dieselbe Zyklenstruktur wie f_{i-1} hat, wird die Berechnung des Normalisators $N_{S_n}(f_1, \ldots, f_{i-1})$ in dieser aufwendigeren Funktion durchgeführt. Hierzu werden die Kandidaten für den Normalisator mit Hilfe von Lemma 4.15 eingeschränkt und diese in der Funktion istNormal() auf Normalität untersucht.

initKonjKlasse()
Bei gegebenem lateinischen Rechteck $(f_1, \ldots, f_{i-1})$ fassen wir in dieser Funktion Permutationen zu geeigneten Klassen zusammen, welche in die i-te Spalte eingesetzt zu isomorphen Rechtecken führen. Diese Klasse zu einer gegebenen Permutation besteht aus der Konjugationsklasse dieser Permutation bezüglich des Normalisators $N_{S_n}(f_1, \ldots, f_{i-1})$, welche nach Lemma 4.9 zu isomorphen Rechtecken führen. Im Falle $i = 3$ können wir mit den Umformungen aus Lemma 4.11 eine noch größere Klasse zusammenfassen, welche sich mit Hilfe von Lemma 4.12 recht gut berechnen läßt.

konjKlasseGueltig()
In dieser Funktion untersuchen wir, ob wir eine Permutation aus der Klasse und damit die ganze Klasse ablehnen können. Dazu überprüfen wir zunächst die Überdeckungsfreiheit bezüglich $f_1, \ldots, f_{i-1}$. Hierbei reicht es, die Überdeckungsfreiheit bei nur einer Permutation der Klasse zu überprüfen, da die verwendeten Umformungen in Lemma 4.9 und Lemma 4.11 aus T'_{i-1} sind und die Überdeckungsfreiheit unter diesen Umformungen nach Lemma 3.5 erhalten bleibt. Als nächstes kontrollieren wir, ob alle Permutationen der Klasse in gueltigPerm[i] sind. Denn ist eine der Permutationen nicht in gueltigPerm[i], so ist die Isomorphieklasse des zugehörigen lateinischen Rechtecks schon behandelt worden und wir können somit die gesamte Klasse von weiteren Betrachtungen ausschließen. Dies ist jedoch nur aussichtsreich, falls

der Normalisator mittels initNormalisator() berechnet wurde, denn wurde der Normalisator mittels initZentralisator() berechnet, so ist normalisator[i] in normalisator[i-1] enthalten. Daher ist für eine Permutation p die Konjugationsklasse K_i bezüglich des Normalisators normalisator[i] in K_{i-1}, der Konjugationsklasse von p bezüglich normalisator[i-1], enthalten. Nun ist die einzige Möglichkeit, daß eine Permutation nicht in gueltigPerm[i] enthalten ist, diejenige, daß sie und ihre Konjugationsklasse aus gueltigPerm[j] mit $j < i$ gelöscht wurde. Ist jedoch eine Permutation von $K_i \subseteq K_{i-1}$ aus gueltigPerm$[i-1]$ gelöscht, so ist die ganze Klasse K_{i-1} als ungültig erkannt worden und inklusive p aus gueltigPerm[i-1] und damit auch aus gueltigPerm[i] gelöscht worden.

konfigAusgabe()
Ist ein lateinisches Quadrat fertiggestellt worden, so wird in dieser Funktion nach einer Anzahl von Aufrufen, welche durch die Konstante anzahlProZeile gegeben ist, das aktuelle Quadrat in der Datei konN_D.aus gespeichert, um bei einem nicht programmgemäßen Abbruch ein Aufsetzen auf das letzte gespeicherte Quadrat zu ermöglichen. Hierbei ist durch N die Ordnung des Quadrates und durch D die Nummer der Datei gegeben, da bei Überschreiten der durch die Konstante zeilenProDatei gegebenen Anzahl von gespeicherten lateinischen Quadraten eine neue Datei angelegt wird. Weiterhin wird in die Datei ausgabe.txt die jeweils bisher benötigte Rechenzeit geschrieben, um Vergleiche verschiedener Algorithmen bezüglich der Rechenzeit zu ermöglichen. Als nächstes legen wir in der Funktion init() der Klasse Matrix die erste Blockzeile unserer doppelgeordneten Inzidenzmatrix durch unser gegebenes lateinisches Quadrat fest und berechnen in der Funktion run() der Klasse Matrix die Inzidenzmatrix nach dem Backtracking-Verfahren fertig.

Das Backtracking-Verfahren

Der zweite Teil unseres Algorithmus, das Backtracking-Verfahren in der Klasse Matrix, orientiert sich an dem Programm positions von M. Mann [19], allerdings mit verbesserter Datenstruktur. Diese Datenstruktur zur Beschreibung eines Zeilenabschnittes, also einer Zeile eines der Permutationsmatrizen des Permutationsteils, wird dabei durch die Klasse MatrixElement bereitgestellt, auf welche wir im folgenden näher eingehen.

MatrixElement
Diese Klasse enthält folgende Objektvariablen:

x, y und z: die Blockzeile, relative Zeile und Blockspalte des jeweiligen Zeilenabschnitts. Diese Koordinaten sind nötig, da wir mit einem Zeiger auf den Matrixelementen wandern und dabei aus Geschwindigkeitsgründen die Koordinaten nicht mitberechnen.

zahlen: Die Stelle des Zeilenabschnitts, an welcher die Eins steht.

bits: Die Stelle des Zeilenabschnitts, an welcher die Eins steht, in Bit-Codierung, d. h. die Zahl bits enthält in Bit-Schreibweise an genau derselben Stelle eine Eins, an der der zugehörige Zeilenabschnitt eine Eins enthält.

zeilen, spalten und blockspalten: Diese Variablen enthalten die Stellen, welche aufgrund der Überdeckungsfreiheit der Blöcke einer Blockzeile, der Blöcke einer Blockspalte und der Zeilenabschnitte in einer Permutationsmatrix durch die schon besetzten Zeilenabschnitte verboten sind, in Bit-Codierung, d. h. die Zahlen in Bit-Schreibweise haben an genau den Stellen eine Eins, welche aufgrund der Überdeckungsfreiheit verboten sind.

vorRechteck: In diesem Array sind in vorRechteck[i] diejenigen Stellen in Bit-Codierung gespeichert, welche aufgrund der Rechtecksregel bezüglich des Zeilenabschnitts i Stellen links des aktuellen Zeilenabschnitts verboten sind.

verboten: Die durch die verschiedenen Regeln verbotenen Stellen in Bit-Codierung.

rechteckPtr: Ein Zeiger auf das Matrixelement in der Blockzeile darüber, welches in der Variablen zahlen mit dem aktuellen Matrixelement übereinstimmt. Hierdurch wird eine effektive Behandlung der Rechtecksregel ermöglicht.

fest: Diese boolesche Variable gibt an, ob das Matrixelement durch die Vorgabe der ersten Blockzeile festgesetzt ist, oder geändert werden darf.

Matrix

In dieser Klasse geschieht die eigentliche Berechnung der Inzidenzmatrix. Der Permutationsteil wird dabei durch das dreidimensionale Array element der Größe $(n-1)\times n\times(n-1)$ beschrieben, gemäß der Anzahl an Blockzeilen, relativen Zeilen und Blockspalten, wobei n die Ordnung der zu berechnenden projektiven Ebene ist. Die erste Blockzeile hiervon wird sogleich in der Funktion initial() durch das in der Klasse Zentral berechnete lateinische Quadrat festgelegt. Der Rest der Matrix wird in der Funktion run() durch das Backtracking-Verfahren ergänzt.

run()

In dieser Funktion läuft der Zeiger iterator auf dem Array element, vorwärts in der Funktion forward(), wobei der letzte Zeilenabschnitt mit dem jeweils kleinsten Eintrag besetzt wird, der sich mit den bisherigen besetzten Zeilenabschnitten widerspruchsfrei einsetzen läßt. Kommt es dabei für alle möglichen Einträge zu einem Widerspruch, so wird in der Funktion back() der

letzte Zeilenabschnitt gelöscht, der Zeiger iterator geht einen Zeilenabschnitt zurück und besetzt diesen mit dem nächstgrößeren Eintrag, der sich widerspruchsfrei setzen läßt. Gelangen wir mit diesem Verfahren zum Ende der Inzidenzmatrix, so speichern wir in der Funktion writeMatrix() den Permutationsteil der Inzidenzmatrix der projektiven Ebene in die Datei „matx_y.pbm“ im pbm-Format, bei dem jede 1 durch ein schwarzes und jede 0 durch ein weißes Kästchen dargestellt wird. Hierbei ist x die Ordnung der projektiven Ebene und y die Nummer der berechneten Matrix.

forward()

Bevor der Eintrag im neuen Zeilenabschnitt festgelegt werden kann, müssen noch die durch die schon gesetzten Zeilenabschnitte verbotenen Stellen berechnet werden. Hierfür ermitteln wir zunächst zeilen, die durch die Überdeckungsfreiheit der Blöcke einer Blockzeile verbotenen Stellen. Dazu fassen wir in der Funktion nextZeile() die Variablen zeilen und bits des Zeilenabschnitts links vom aktuellen Zeilenabschnitt, falls vorhanden, mit der bitweisen Oder-Verknüpfung zusammen und speichern das Ergebnis in zeilen des aktuellen Zeilenabschnitts. Auf ähnliche Weise setzen wir auch blockSpalten und spalten, die verbotenen Stellen aufgrund der Überdeckungsfreiheit der Zeilenabschnitte in einer Permutationsmatrix und der Überdeckungsfreiheit der Blöcke einer Blockspalte. Aufwendiger ist die Berechnung der Verbote aufgrund der Rechtecksregel. Hierzu wird in vorRechteck[i] jeweils gespeichert, welche Einträge verboten sind aufgrund der Rechtecksregel mit dem Zeilenabschnitt, welcher i Blockspalten links vom aktuellen Zeilenabschnitt ist, falls noch in derselben Zeile. Weiterhin zeigt rechteckPtr jeweils auf den Zeilenabschnitt in der Blockzeile darüber, falls vorhanden, und derselben Blockspalte, welcher denselben Eintrag wie der aktuelle Zeilenabschnitt enthält. Dann können wir vorRechteck[i] des aktuellen Zeilenabschnitts berechnen, indem wir von der aktuellen Position i Zeilenabschnitte nach links gehen, dann mithilfe des Zeiger rechteckPtr zu dem Zeilenabschnitt in der Blockzeile darüber, welcher denselben Eintrag hat, von diesem wieder i Zeilenabschnitte nach rechts, so daß wir wieder in derselben Blockspalte sind wie unser anfänglicher Zeilenabschnitt. Dann sind vorRechteck[i] zusammen mit bits dieses Zeilenabschnitts die Verbote für vorRechteck[i] des aktuellen Zeilenabschnitts. Diese ganzen Verbote werden dann in der Funktion setzeVerboten() mit bitweisem Oder verknüpft und in verboten gespeichert. Nun können wir dank unserer Bit-Schreibweise jeden Eintrag mit nur einer Operation auf Gültigkeit bezüglich den Überdeckungsfreiheiten und der Rechtecksregel überprüfen, indem wir ihn in Bit-Schreibweise mit bitweisem Und mit verboten verknüpfen. Ist das Ergebnis dabei ungleich 0, so verstößt der Eintrag gegen eine dieser Regeln. Bei 0 beginnend wird nun der Eintrag erhöht, bis es das erste mal zu keinem Widerspruch kommt.

back()

Gehen wir einen Zeilenabschnitt zurück, so sind die Verbote in der Varia-

blen verbote immer noch gültig und braucht daher nicht neu berechnet zu werden. Nun erhöhen wir wiederum den Eintrag so lange, bis es erneut zu keinem Widerspruch kommt.

Alleine die verbesserte Datenstruktur des Backtracking-Verfahrens verkürzt die Rechenzeit im Vergleich zum Programm positions von M. Mann [19] immerhin auf die Hälfte. Der weitaus größere Geschwindigkeitsgewinn folgt jedoch aus der stärkeren Einschränkung der ersten Blockzeile. Weiterhin zeigt sich daß die Anfangsquadrate durch diesen Algorithmus bei weitem noch nicht auf einen Repräsentanten je Isomorphieklasse reduziert werden, so daß hier noch weiteres Beschleunigungspotential vorhanden zu sein scheint. Bei ersten Versuchen mit Verfahren zur weiteren Einschränkung der Anfangsquadrate stellten sich diese Verfahren jedoch als so aufwendig heraus, daß die hierfür benötigte Rechenzeit die eingesparte Rechenzeit beim Backtracking-Verfahren weit übertrifft, so daß die gesamte Rechenzeit sogar ansteigt.

Beim Algorithmus zur Reduzierung der Anfangsquadrate zeigte sich außerdem, daß die Berechnung des Normalisators mit der Funktion initNormalisator() gegenüber der schnelleren Funktion initZentralisator(), bei der nur der Zentralisator berechnet wird, die Zahl der Anfangsquadrate nur bis zur dritten Spalte effektiv reduziert, während bei späteren Spalten hierdurch die Anzahl der Anfangsquadrate nur unwesentlich verringert wird. Somit kann ab der vierten Spalte beruhigt nur noch die schnellere Funktion initZentralisator() verwendet werden.

Von großer Bedeutung für die Anzahl der berechneten lateinischen Quadrate ist dagegen die Ordnung auf den Zyklenstrukturen, wodurch auch die Ordnung auf den lateinischen Rechtecken wesentlich mitbestimmt wird. Hierbei zeigt sich die Reduktion der Quadrate dann am effektivsten, wenn die Zyklenstrukturen nach der Größe ihres Zentralisators geordnet sind, wobei die kleinste Zyklenstruktur den größten Zentralisator hat.

Für unsere Berechnungen verwendeten wir als Compiler den GNU-C++-Compiler und führten das Programm unter Linux auf einem Pentium IV-Prozessor mit $2,4$ GHz aus. Dabei wird die (optimierte) Compilierung durch den Befehl ,,g++ Main.cpp -O“ aufgerufen und das ausführbare Programm in der Datei ,,a.out“ gespeichert. Leider benötigt die Messung der Rechenzeit in der Bibliothek getclock.h einige spezielle Funktionen von Linux, so daß die Compilierung unter einer anderen Plattform (zumindest unter Beibelassung der Zeitmessung) unter Umständen nicht funktioniert. Das Programm läßt sich einfach durch Aufruf von ,,a.out“ starten und liest dann von den Dateien ,,konx_y.aus“ diejenige ein, welche mit größtem y existiert. Daraufhin wird auf die Konfiguration, welche durch die letzte Zeile der Datei gegeben ist, aufgesetzt und die restlichen Inzidenzmatrizen berechnet. Existiert keine dieser Dateien oder wird das Programm per ,,a.out neu“ aufgerufen, so beginnt das Programm am Anfang. Weitere interessante Daten, wie die

bisherige Rechenzeit bei jedem Zwischenspeichern eines lateinischen Quadrates, werden in der Datei „ausgabe.txt“ gespeichert. In der folgenden Tabelle listen wir zu jeder Ordnung der projektiven Ebene die dabei benötigte Rechenzeit, die Anzahl der berechneten Inzidenzmatrizen und den Faktor, um den die Rechenzeit gegenüber der Berechnung der Inzidenzmatrizen der nächstkleineren Ordnung ansteigt, auf.

Ordnung	Rechenzeit	Anzahl der Inzidenzmatrizen	Faktor
3	$0,00$ s	1	
4	$0,00$ s	1	
5	$0,00$ s	1	
6	$0,01$ s	0	
7	$0,28$ s	1	28
8	$29:08$ min	8	$6,2 \cdot 10^3$

Bei den kleinen Ordnungen sind die Laufzeiten natürlich wenig aussagekräftig, da sie von der Meßungenauigkeit von $0,01$ sec überlagert werden, doch zeigt sich spätestens bei Ordnung 8 mit einer Laufzeit von einer knappen halben Stunde ein deutlicher Fortschritt gegenüber dem Programm von M. Mann [19], welches auf demselben Rechner noch zwei Tage benötigte. Dennoch reicht dieser Geschwindigkeitsgewinn noch nicht für eine vollständige Behandlung der Ordnung 9 aus.

4.5 Weitere Umformungen doppelgeordneter Inzidenzmatrizen

Wir verbessern nun unseren Algorithmus durch folgende Überlegungen. Wir nehmen vom Permutationsteil

$$P = \begin{pmatrix} P_{1,1} & \cdots & P_{1,n-1} \\ \vdots & \ddots & \vdots \\ P_{n-1,1} & \cdots & P_{n-1,n-1} \end{pmatrix}$$

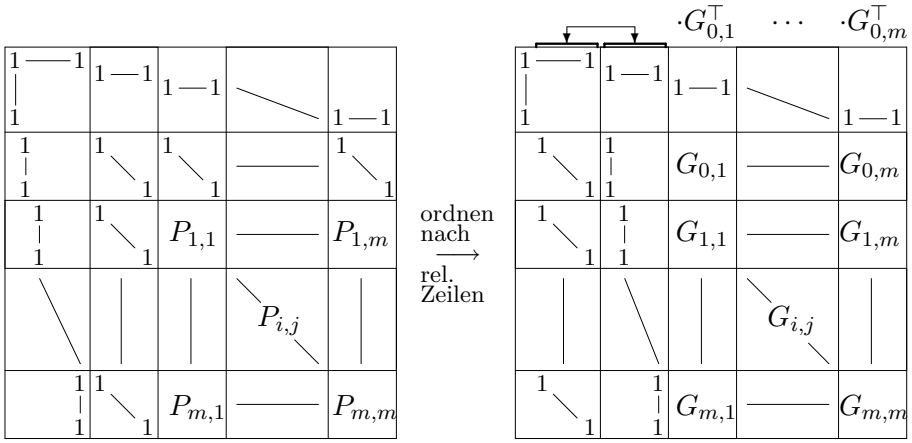
jeweils die ersten relativen Zeilen und schreiben mit unserer Transformation ein Quadrat

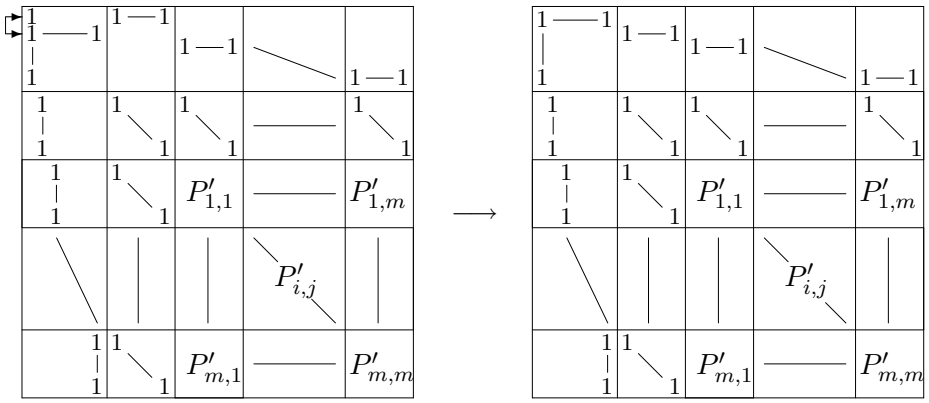
$$A_1 = \begin{pmatrix} \tilde{p}_{1,1}(1) & \tilde{p}_{1,2}(1) & \cdots & \tilde{p}_{1,n-1}(1) \\ \tilde{p}_{2,1}(1) & \tilde{p}_{2,2}(1) & \cdots & \tilde{p}_{2,n-1}(1) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{p}_{n-1,1}(1) & \tilde{p}_{n-1,2}(1) & \cdots & \tilde{p}_{n-1,n-1}(1) \end{pmatrix}$$

Dabei sind die Permutationen $\tilde{p}_{i,j}$ durch die Permutationsmatrizen $P_{i,j}$ durch $\tilde{p}_{i,j}(a) = b :\Leftrightarrow e_a^\top P_{i,j} = e_b^\top$ definiert. Dann ist A_1 ein lateinisches Quadrat über den Zahlen $2, \ldots, n$, denn 1 kommt nicht vor aufgrund der Fixpunktfreiheit, und in jeder Zeile und jeder Spalte kommen die Zahlen $2, \ldots, n$ jeweils nur einmal vor aufgrund der horizontalen und vertikalen Überdeckungsfreiheit. Allgemeiner bilden die j-ten relativen Zeilen mit dieser Umformung ein lateinisches Quadrat A_j der Ordnung $n-1$ über den Zahlen $1, \ldots, j-1, j+1, \ldots, n$. Dieses nennen wir dann **j-tes reduziertes lateinisches Quadrat**.

Ziel ist es nun, auch die ersten reduzierten lateinischen Quadrate zu Klassen zusammenzufassen, bei denen es ausreicht, jeweils nur einen Repräsentanten einzusetzen, ohne zu riskieren, eine Isomorphieklasse von projektiven Ebenen zu verlieren. Hierzu nutzen wir Permutationsäquivalenzen, die die Doppelordnung erhalten. Zunächst sieht man sofort, daß sich ein Permutieren der Zeilen bzw. Spalten des lateinische Quadrates A_1 durch ein geeignetes Permutieren der inneren Blockzeilen bzw. Blockspalten erreichen läßt. Das Permutieren der Nummerierung des lateinischen Quadrates A_1 bewirkt man durch Konjugation aller Permutationsmatrizen des Permutationsteils mit einer Permutationsmatrix W, welche e_1 als Fixpunkt hat. Hierdurch bleiben die relativen ersten Zeilen an ihrem Ort bei der Linksmultiplikation mit W, da W als Fixpunkt e_1 hat, und die Rechtsmultiplikation mit $W^{-1} = W^\top$ verursacht eine Permutation der Nummerierung. Da A_1 nur die Zahlen $2, \ldots, n$ enthält, muß auch deswegen W^{-1}, und damit auch W, als Fixpunkt e_1 haben. Die gesamte Umformung läßt sich dabei durch den permutationsäquivalenten Übergang der Inzidenzmatrix I zu $\operatorname{diag}(E_{n+1}, W, \ldots, W)\, I \operatorname{diag}(E_{n+1}, W, \ldots, W)^\top$ beschreiben.

Insgesamt können wir uns nach diesen bisherigen Umformungen somit auch bei der Vorgabe des Quadrates A_1 auf einen Repräsentanten aus jeder Familie von lateinischen Quadraten der Ordnung $n-1$ beschränken. Weitere Einschränkungen erreichen wir durch Vertauschen der Blockzeilen und relativen Zeilen des Kerns, wodurch sich folgende Konstellation ergibt:

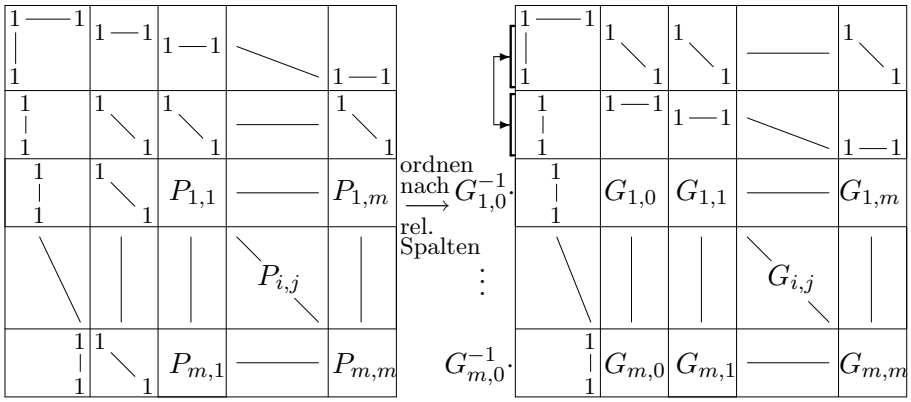


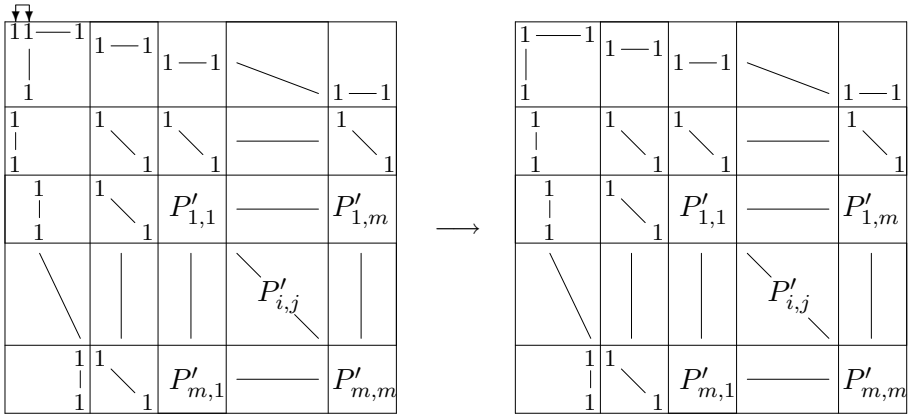


Hierbei entstehen die Permutationsmatrizen $G_{i,j}$ aus den $P_{i,j}$ durch Sortieren nach den relativen Zeilen, d. h. es gilt $e_k^\top G_{i-1,j} = e_i^\top P_{k-1,j}$ für $i,k = 1,\ldots,n$ und $j = 1,\ldots,m = n-1$, wobei wir wieder $P_{0,j} = E_n$ setzen. Insbesondere ist $e_1^\top G_{i-1,j} = e_i^\top P_{0,j} = e_i^\top E_n = e_i^\top$ und $e_i^\top G_{0,j} = e_1^\top P_{i-1,j} = \begin{cases} e_1^\top & \text{für } i = 1 \\ e_{A_1(i-1,j)}^\top & \text{sonst} \end{cases}$ für alle $i = 1,\ldots,n$ und $j = 1,\ldots,n-1$, wobei A_1 das erste reduzierte lateinische Quadrat ist. Zudem ist $P'_{i,j} = G_{i,j}G_{0,j}^\top = G_{i,j}G_{0,j}^{-1}$. Diese Permutationsäquivalenz läßt sich als Abbildung $I \mapsto \operatorname{diag}(W_2, E_{n-1}, W_3)\, I \operatorname{diag}(E_1, W_1, G_{0,1}, \ldots, G_{0,m})^\top$ beschreiben, wobei $W_1 = \begin{pmatrix} & E_n \\ E_n & \end{pmatrix} \in \mathcal{S}_{2n}$ und $W_2 = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix} \in \mathcal{S}_2$ sind, und $W_3 \in \mathcal{S}_{n^2}$

definiert ist durch $(W_3)_{i,j} = \begin{cases} 1 & \text{für } (i,j) = (na+b+1, nb+a+1) \\ & \text{mit geeigneten } a,b \in \{0,\ldots,n-1\} \\ 0 & \text{sonst.} \end{cases}$

Bezeichnen wir nun mit A_1' das erste reduzierte lateinische Quadrat der transformierten Inzidenzmatrix. Dann gilt $A_1'(i,j) = k+1 \Leftrightarrow e_1^\top P_{i,j}' = e_1^\top G_{i,j} G_{0,j}^{-1} = e_{k+1}^\top \Leftrightarrow e_{i+1}^\top = e_1^\top G_{i,j} = e_{k+1}^\top G_{0,j} = e_1^\top P_{k,j} \Leftrightarrow A_1(k,j) = i+1$ für alle $i,j,k = 1,\ldots,n-1$. Es entsteht A_1' aus A_1 also durch Vertauschen von Zeile und Nummerierung, wobei die Indexverschiebung $i \to i+1$ bzw. $k \to k+1$ daher rührt, daß A_1 bzw. A_1' lateinische Quadrate über den Zahlen $2,\ldots,n$ sind. Eine ähnliche Umformung erhalten wir durch Vertauschen von Blockspalten und relativen Spalten wie folgt:





Diese Permutationsäquivalenz läßt sich durch die Umformung der Inzidenzmatrix I zu $\operatorname{diag}(E_1, W_1, G_{1,0}^{-1}, \ldots, G_{m,0}^{-1})\, I \operatorname{diag}(W_2, E_{n-1}, W_3)^\top$ beschreiben, wobei W_1, W_2 und W_3 wie zuvor definiert sind. Die $G_{i,j}$ entstehen hierbei aus den $P_{i,j}$ durch Sortieren nach den relativen Spalten, d. h. es gilt $e_l^\top G_{i,j-1} = e_k^\top \Leftrightarrow e_l^\top P_{i,k-1} = e_j$ für $j,k,l = 1,\ldots,n$ und $i = 1,\ldots,m = n-1$, wobei $P_{i,0} = E$ für alle i. Insbesondere ist $e_1^\top G_{i,0} = e_1^\top$, also auch

$e_1^\top G_{i,0}^{-1} = e_1^\top G_{i,0}^\top = e_1^\top$, da $e_1^\top P_{i,0} = e_1^\top E = e_1^\top$. Weiterhin gilt $P'_{i,j} = G_{i,0}^{-1} G_{i,j}$. Ist A_1 das erste reduzierte lateinische Quadrat der ursprünglichen Inzidenzmatrix und A'_1 das erste reduzierte lateinische Quadrat der transformierten Matrix, so gilt $A_1(i,j) = k+1 \Leftrightarrow e_1^\top P_{i,j} = e_{k+1}^\top \Leftrightarrow e_1^\top G_{i,k} = e_{j+1}^\top \Leftrightarrow e_1^\top P'_{i,k} = e_1^\top G_{i,0}^{-1} G_{i,k} = e_1^\top G_{i,k} = e_{j+1}^\top \Leftrightarrow A'_1(i,k) = j+1$ für alle $i,j,k = 1,\ldots,m = n-1$. Dies besagt jedoch gerade, daß A'_1 aus A_1 durch Vertauschen von Spalte und Nummerierung entsteht. Der Übergang $i \to i+1$ bzw. $k \to k+1$ liegt wiederum nur daran, daß A'_1 bzw. A_1 lateinische Quadrate über den Zahlen $2,\ldots,n$ sind. Insgesamt können wir also noch die Spalten mit der Nummerierung und die Zeilen mit der Nummerierung vertauschen. Dann können wir uns bei der Vorgabe des ersten reduzierten lateinischen Quadrates jedoch sogar auf einen Repräsentanten aus jeder Isomorphieklasse beschränken. Dies bringt eine starke Beschleunigung unseres Algorithmus mit sich, da wir als erste Vorgabe nicht mehr die Isomorphieklassen von lateinischen Quadraten der Ordnung n sondern nur der Ordnung $n-1$ haben. Zwar dauert das Backtracking-Verfahren nach der Vorgabe der relativen ersten Zeilen länger als wenn wir eine ganze erste Blockzeile vorgeben, jedoch wird das bei Weitem dadurch kompensiert, daß die Anzahl der Isomorphieklassen lateinischer Quadrate von Ordnung $n-1$ auf Ordnung n so stark ansteigt. Denn ist die Situation bei Ordnung 2 und 3 mit je einer und bei Ordnung 4 und 5 mit je zwei Isomorphieklassen noch recht übersichtlich, gibt es schon 12 Isomorphieklassen der Ordnung 6. Bei Ordnung 7 steigt deren Anzahl auf 147 an und für Ordnung 8 gibt es gar 283.657 Isomorphieklassen von lateinischen Quadraten.

4.6 das Programm-Paket NeuInzidenz

Zur Implementierung dieses neuen Algorithmus können wir viele Teile in leicht veränderter Form vom Programm-Paket Inzidenzmatrix wiederverwenden. Dies drückt sich auch dadurch aus, daß die meisten Klassen- und Dateinamen gleich sind bis auf jeweils die Vorsilbe „neu“. Gestartet wird das Programm wiederum in der Datei „Main.cpp“, in der auch die Dateien „neuOrdnung3.h, ..., neuOrdnung10.h“ für die Festlegung der Ordnung der projektiven Ebene eingebunden werden. Dabei ist wieder DIM die Ordnung n der projektiven Ebene. Die Ordnung des zu berechnenden lateinischen Quadrats ist jedoch um eins kleiner als n und wird in der Variablen LATDIM gespeichert. Im Unterschied zu den alten Dateien sind jedoch die Zyklenstrukturen der fixpunktfreien Permutationen der Länge LATDIM und nicht DIM hierin angegeben. Die Reduzierung der Zahl der lateinischen Quadrate der Ordnung LATDIM geschieht wieder in der Klasse NeuZentral in der Datei „neuZentral.h“. Ebenfalls werden als Hilfsklassen zur Beschreibung der Permutationen die Klassen NeuPerm und NeuSimpelPerm in den Dateien „neuPerm.h“ und „neuSimpelPerm.h“ verwandt. Die Berechnung der Inzidenzmatrix aus den vorgegebenen ersten relativen Zeilen wird in der Klasse Neu-

Matrix, welche in der Datei „neuMatrix.h“ implementiert ist, durchgeführt. Hierbei hat es sich als vorteilhaft erwiesen, beim Backtracking-Verfahren die Zeilen nicht in der üblichen Reihenfolge abzuarbeiten, sondern, da ja alle ersten relativen Zeilen vorgegeben sind, auch weiterhin jeweils die Zeilen mit gleichen relativen Zeilennummern gemeinsam zu berechnen. Die Hilfsklasse MatrixElement mußte nicht verändert werden und behielt daher auch ihren alten Namen.

Hierbei dauert die Berechnung der Inzidenzmatrizen bei vorgegebenen ersten relativen Zeilen deutlich länger als bei vorgegebener erster Blockzeile. Daher können wir die Rechenzeit verkürzen, indem wir aufwendigere Verfahren zur Reduzierung der Zahl der Anfangsquadrate verwenden, Verfahren, welche die Rechenzeit bei unserem alten Programm noch erhöhten. Besonders hilfreich ist dabei das folgende kleine Lemma, durch welches wir gewisse lateinische Rechtecke durch einen Schnelltest ausschließen können.

Lemma 4.16
Seien $F := (f_1, \ldots, f_m)$ *und* $G := (g_1, \ldots, g_m)$ *zwei lateinische Rechtecke, wobei* $f_1 = \mathrm{id} = g_1$. *Ist* $\mathrm{Zykl}(f_2) < \mathrm{Zykl}(g_2)$, *so ist* $F < G$. *Sind* $\mathrm{Zykl}(f_2) = \mathrm{Zykl}(g_2)$ *und* $\mathrm{Zykl}(f_3) < \mathrm{Zykl}(g_3)$, *so gibt es ein zu* F *isomorphes lateinisches Rechteck* $\tilde{F}$ *mit* $\tilde{F} < G$.

Beweis:

Seien F und G lateinische Rechtecke wie im Lemma vorgegeben. Sei zunächst $\mathrm{Zykl}(f_2) < \mathrm{Zykl}(g_2)$, dann ist nach unserer Zyklenstrukturordnung $f_2 < g_2$. Da weiterhin $f_1 = \mathrm{id} = g_2$, folgt hieraus $F < G$.
Seien also nun $f_1 = \mathrm{id} = g_1$, $\mathrm{Zykl}(f_2) = \mathrm{Zykl}(g_2)$ und $\mathrm{Zykl}(f_3) < \mathrm{Zykl}(g_3)$. Da f_2 und g_2 gleiche Zyklenstrukturen haben, sind sie konjugiert zueinander, es gibt also ein $t \in S_n$ mit $t\, f_2\, t^{-1} = g_2$. Dann ist $\tilde{F} = (\tilde{f}_1, \ldots, \tilde{f}_m) := (t\, f_1\, t^{-1}, \ldots, t\, f_m\, t^{-1}) = (g_1, g_2, t\, f_3\, t^{-1}, \ldots, t\, f_m\, t^{-1})$ isomorph zu F mit $\tilde{f}_1 = \mathrm{id} = g_1$, $\tilde{f}_2 = g_2$ und $\tilde{f}_3 < g_3$, da $\mathrm{Zykl}(\tilde{f}_3) = \mathrm{Zykl}(f_3) < \mathrm{Zykl}(g_3)$. Also ist $\tilde{F} < G$.

□

Dieses Lemma können wir als Schnelltest nutzen, wenn wir $G := (g_1, \ldots, g_m)$ als das aktuelle lateinische Rechteck ansehen und $F := (f_1, \ldots, f_m)$ als ein Rechteck, welches aus G durch isomorphe Umformungen entsteht, wobei die Rechtecke zu $f_1 = \mathrm{id} = g_1$ normiert sind. Betrachten wir nun die Zyklenstrukturen der übrigen Permutationen von F und G. Ist dabei die kleinste Zyklenstruktur von F kleiner als die kleinste Zyklenstruktur von G, oder aber sind die kleinsten Zyklenstrukturen von F und G gleich und die nächstgrößere Zyklenstruktur von F kleiner als die nächstgrößere Zyklenstruktur von G, so gibt es ein zu F und damit auch G isomorphes Rechteck $\tilde{F}$, welches kleiner ist als G. Damit ist G nicht das kleinste Rechteck in seiner Isomorphieklasse, wurde also schon behandelt und kann daher von weiteren Berechnungen ausgeschlossen werden. Das Wichtige bei diesem Test ist, daß

wir nur die Zyklenstrukturen der Permutationen berechnen und vergleichen müssen, wobei dies sehr viel schneller geht, als F durch andere isomorphe Umformungen in ein kleineres Rechteck als G abzubilden. Leider ist dieser Test auch nicht sehr genau, so daß viele Rechtecke, welche durch dieses Verfahren nicht ausgeschlossen werden können, durch aufwendigere Verfahren sehr wohl als redundant erkannt werden.
Dieses Verfahren nutzen wir bei der Erweiterung der Funktion konjKlasseGueltig():

konjKlasseGueltig()

Diese Funktion, welche schon im Programm-Paket Inzidenz erläutert wurde, wird im folgenden erweitert, um weitere isomorphe Rechtecke für die späteren Berechnungen zusammenzufassen. Hierzu multiplizieren wir das aktuelle lateinische Rechteck $G = (g_1 = \mathrm{id}, g_2, \ldots, g_m)$ von links mit g_j^{-1}, mit geeignetem $j \in \{2, \ldots, m\}$, und vertauschen die erste und die j-te Spalte, was das Rechteck $F = (f_1, \ldots, f_m) := (\mathrm{id}, \ldots, g_j^{-1}g_{j-1}, g_j^{-1}, g_j^{-1}g_{j+1}, \ldots, g_j^{-1}g_m)$ ergibt. Hierbei ist F zu G isomorph und zu $f_1 = \mathrm{id} = g_1$ normiert. Nun wenden wir zunächst unseren Schnelltest nach Lemma 4.16 an und überprüfen, ob wir G als nicht minimal in seiner Isomorphieklasse von weiteren Berechnungen ausschließen können. War dies nicht erfolgreich, so versuchen wir in einem ungleich aufwendigeren Verfahren, F per Spaltenvertauschung und durch Konjugation der durch die Spalten von F beschriebenen Permutationen in ein Rechteck $F' = (f'_1, \ldots, f'_m)$ umzuformen mit $f'_1 = g_1, \ldots, f'_{r-1} = g_{r-1}$ für ein $r \leq m$. Ist nun f'_r nicht in gueltigPerm[r] enthalten, so sind bereits alle lateinischen Rechtecke, welche $(f'_1, \ldots, f'_r)$ als Teilrechteck haben, behandelt worden. Insbesondere ist die Isomorphieklasse von F', also auch G, schon behandelt worden und kann daher von weiteren Berechnungen ausgeschlossen werden.

endGueltig()

Eine isomorphe Umformung der lateinischen Quadrate haben wir bisher noch gar nicht beachtet, nämlich die Vertauschung von Zeilen und Spalten, also das Transponieren des Quadrates. Dies liegt einfach daran, daß diese Umformung nur für ein bereits vollständiges lateinisches Quadrat möglich ist und wir somit bestenfalls auch nur dieses eine lateinische Quadrat ausschließen können. Daher muß der verwendete Algorithmus zur Überprüfung des Quadrates wesentlich schneller sein, als das Fertigrechnen der Inzidenzmatrix nach dem Backtracking-Verfahren. Hier leistet unser Schnelltest nach Lemma 4.16, diesmal auf das transponierte Quadrat angewandt, gute Dienste und wird in dieser Funktion bei jedem fertigen Quadrat durchgeführt.

Die Funktionalität bei Compilierung und Aufruf des Programmes gleicht der des vorigen Programms, jedoch werden, um Verwirrungen mit dem alten Programm zu vermeiden, die Dateien zum Zwischenspeichern der ersten

reduzierten lateinischen Quadrate „konfx_y.aus“ genannt, wobei wiederum x die Ordnung der projektiven Ebene und y die Nummer der Datei sind. Dieser neue Algorithmus erfährt gegenüber dem alten einen deutlichen Geschwindigkeitszuwachs, so daß nun auch Ordnung 9 in akzeptabler Zeit behandelt werden kann. In der folgenden Tabelle listen wir wiederum zu jeder Ordnung der projektiven Ebene die dabei benötigte Rechenzeit, die Anzahl der berechneten Inzidenzmatrizen und den Faktor, um den die Rechenzeit gegenüber der Berechnung der Inzidenzmatrizen der nächstkleineren Ordnung ansteigt, auf.

Ordnung	Rechenzeit	Anzahl der Inzidenzmatrizen	Faktor
4	$0,00$ s	1	
5	$0,00$ s	2	
6	$0,00$ s	0	
7	$0,02$ s	2	
8	$11,9$ s	2	$6 \cdot 10^2$
9	33 d	367	$2 \cdot 10^5$

Auch der deutliche Geschwindigkeitsgewinn durch diesen neuen Algorithmus reicht jedoch noch nicht aus, um Ordnung 10 in Angriff nehmen zu können. Weiterhin wäre natürlich ein Vergleich mit dem Algorithmus von C. W. H. Lam [15] zur Berechnung der projektiven Ebenen der Ordnung 9 von besonderen Interesse. Leider gibt er in seinem Artikel jedoch keine Hinweise auf die benötigte Rechenzeit und den verwendeten Computer. Interessanterweise beginnt auch er mit einem Repräsentantensystem der Isomorphieklassen der lateinischen Quadrate der Ordnung 8. Dieses Repräsentantensystem reduziert er dabei auf tatsächlich einen Repräsentanten je Isomorphieklasse, wodurch sich 283.657 Anfangsquadrate ergeben. Im Vergleich dazu reduziert unser Algorithmus in NeuInzidenz die lateinischen Quadrate der Ordnung 8 nur auf 1,1 Mio. Anfangsquadrate, also um einen Faktor 4 mehr Anfangsquadrate. Dennoch berechnet unser Algorithmus mit 367 Inzidenzmatrizen nur unwesentlich mehr als die 325 Matrizen von Lam. Diese große Anzahl an Inzidenzmatrizen für Ordnung 9 stellt uns jedoch vor ein anderes Problem, nämlich welche von ihnen zu derselben projektiven Ebene gehören, also welche der Inzidenzmatrizen permutationsäquivalent sind. Mit diesem Problem beschäftigen wir uns daher im kommenden Kapitel näher.

5 Permutationsäquivalenz von Inzidenzmatrizen projektiver Ebenen

Mit unserem Programm-Paket NeuInzidenz werden ab Ordnung 5 mehrere Inzidenzmatrizen berechnet. Damit stellt sich die Frage nach der Isomorphie der zugehörigen Ebenen. Formuliert für Inzidenzmatrizen heißt dies: Gibt es zu Inzidenzmatrizen I_1 und I_2 Permutationsmatrizen P und Q, so daß $PI_1Q^\top = I_2$. Dies bezeichneten wir mit dem Begriff Permutationsäquivalenz. Besonders interessant wird diese Frage bei Ordnung 9, denn während es bis Ordnung 8 immer nur einen Isomorphietyp von projektiven Ebenen, nämlich die desarguesschen Ebenen, gibt, kommen nun erstmals mehrere Isomorphietypen, insbesondere nicht-desarguessche, vor. Zudem erhalten wir bis zu Ordnung 8 nur jeweils höchstens 2 Inzidenzmatrizen und die Isomorphiefrage ist leicht entscheidbar. Bei Ordnung 9 dagegen gibt es mit unserer Vorgehensweise 367 Inzidenzmatrizen, die sich ohne Computerhilfe nicht mehr den 4 bekannten Isomorphieklassen zuordnen lassen.

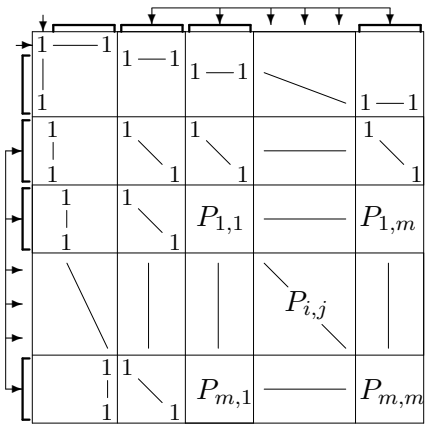
Bereits vorhanden war das Programm pe von B. Schwarz [23] zur Entscheidung der Permutationsäquivalenz bei beliebigen $(0,1)$-Matrizen. Die Funktionsweise des Programms beruht darauf, daß man bei $i = 1$ anfangend immer $e_i^\top P$, also diejenige Zeile, welche durch die Permutationsäquivalenz zur i-ten Zeile werden soll, vorgibt und schaut, ob es eine Permutation Q gibt, so daß die ersten i Zeilen von $PI_1Q^\top$ und I_2 übereinstimmen. Gibt es ein solches Q, so wird der Index i erhöht und dies mit dem neuem Index wiederholt. Gibt es kein solches Q, so wird ein anderes $e_i^\top P$ vorgegeben und damit weitergerechnet. Führen alle möglichen $e_i^\top P$ zu Widersprüchen, so wird der Index i um Eins erniedrigt und dann ein neues $e_i^\top P$ festgesetzt. Dieses Programm wurde auch bei den von M. Mann [19] gefundenen Inzidenzmatrizen verwandt, insbesondere den 24 Inzidenzmatrizen der Ordnung 9. Diese konnten in 4 Klassen von jeweils zueinander permutationsäquivalenten Matrizen eingeteilt werden, doch konnte die Permutationsäquivalenz der verschiedenen Klassen nicht ausgeschlossen werden. Außerdem dauerte der Nachweis der Permutationsäquivalenz zweier Inzidenzmatrizen mit diesem Programm noch durchschnittlich 2 Wochen auf einem Pentium III-Prozessor mit 550 MHz. Diese noch verhältnismäßig kurzen Rechenzeiten ergaben sich jedoch nur, da es zu diesen Inzidenzmatrizen jeweils eine Permutationsäquivalenz gab, die relativ bald getestet wurde. Ein Test an zweien, wie sich später herausstellte, permutationsäquivalenten Inzidenzmatrizen, welche durch das Programm-Paket NeuInzidenz für Ordnung 9 erstellt wurden, brachte auch nach Monaten Rechenzeit keine Ergebnisse. Daher war ein neues, effektiveres Programm zur Untersuchung der Permutationsäquivalenz unabdingbar. Dies ist natürlich nur aussichtsreich, wenn wir spezielle Eigenschaften der Doppelordnung bei projektiven Ebenen ausnutzen.

5.1 Ein neuer Algorithmus

Die grundlegende Idee unseres Programms ist, daß die Blockstruktur einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene im Wesentlichen bereits durch die Festsetzung der ersten Zeile und ersten Spalte vorgegeben ist. Genauer gilt folgendes Lemma:

Lemma 5.1
Seien I und $\tilde{I}$ zwei permutationsäquivalente doppelgeordnete Inzidenzmatrizen einer projektiven Ebene der Ordnung n, wobei $\tilde{I} = P\,I\,Q^\top$ mit Permutationsmatrizen P und Q, für die $Pe_1 = e_1$ und $Qe_1 = e_1$ gilt, d. h. die erste Zeile und erste Spalte werden durch die Permutationsäquivalenz festgehalten. Dann ist $P = \mathrm{diag}(E_1, P_1, \hat{P}_2)$ und $Q = \mathrm{diag}(E_1, P_2, \hat{P}_1)$, wobei $P_1, P_2 \in \mathcal{S}_n$ sind und $\hat{P}_i = (P_i \otimes E_n) \cdot \mathrm{diag}(W_1^i, \ldots, W_n^i) \in \mathcal{S}_{n^2}$ mit $W_1^i, \ldots, W_n^i \in \mathcal{S}_n$ für $i = 1, 2$. Die Kerne unterscheiden sich also nur durch Vertauschen der Blockzeilen bzw. Blockspalten und geeignete Permutation der Zeilen innerhalb einer Blockzeile bzw. Permutation der Spalten innerhalb einer Blockspalte.

Beweis:
Wie in Kapitel 2 bezeichnen wir mit l_i die i-te Zeile und mit p_i die i-te Spalte der Inzidenzmatrix I. Da p_1 durch die Permutationsäquivalenz festgehalten wird und $\tilde{I}$ wiederum doppelgeordnet sein soll, können die Zeilen $l_1, \ldots, l_{n+1}$ nur untereinander permutiert werden. Da weiterhin auch l_1 fixiert wird, müssen sogar $l_2, \ldots, l_{n+1}$ untereinander permutiert werden. Dies kann man durch eine Multiplikation mit einer Permutationsmatrix $P_1 \in \mathcal{S}_n$ von links beschreiben. Somit ist $P = \mathrm{diag}(E_1, P_1, \hat{P}_2)$ mit geeigneten Permutationsmatrizen $P_1 \in \mathcal{S}_n$ und $\hat{P}_2 \in \mathcal{S}_{n^2}$. Mit analoger Argumentation hat Q die Form $Q = \mathrm{diag}(E_1, P_2, \hat{P}_1)$ mit geeigneten Permutationsmatrizen $P_2 \in \mathcal{S}_n$ und $\hat{P}_1 \in \mathcal{S}_{n^2}$. Weiterhin, da die i-te Blockspalte dadurch festgelegt ist, daß dies alle Spalten außer p_1 sind, welche auf der Geraden l_{i+2} liegen für $i = 0, \ldots, n-1$, und die Zeilen $l_2, \ldots, l_{n+1}$ nur untereinander permutiert werden, werden die Blockspalten unter dieser Permutationsäquivalenz wiederum auf Blockspalten abgebildet. Dabei wird die i -te Blockspalte genau dann auf die j-te Blockspalte abgebildet, wenn l_{i+2} auf l_{j+2} abgebildet wird. Dazu können noch Permutationen der Spalten innerhalb einer Blockspalte hinzukommen, wobei jedoch sichergestellt sein muß, daß die erhaltene Inzidenzmatrix wieder doppel-



geordnet ist. Daher muß die Permutationsmatrix $\hat{P}_1$ die Form haben, daß sie aus P_1 entsteht, indem die Einsen durch geeignete Permutationsmatrizen der Größe n ersetzt werden und die Nullen durch die Nullmatrix der Größe n. Mithilfe des Kronecker-Produkts läßt sich dies ausdrücken durch $\hat{P}_1 = (P_1 \otimes E_n) \cdot \mathrm{diag}(W_1^1, \ldots, W_n^1)$ mit geeigneten $W_1^1, \ldots, W_n^1 \in \mathcal{S}_n$. Analog muß $\hat{P}_2 = (P_2 \otimes E_n) \cdot \mathrm{diag}(W_1^2, \ldots, W_n^2)$ mit geeigneten $W_1^2, \ldots, W_n^2 \in \mathcal{S}_n$ sein.

□

Halten wir bei der Permutationsäquivalenz zusätzlich noch die zweite Zeile und Spalte fest, so lassen sich die Permutationsmatrizen noch weiter einschränken. Das führt zu folgendem Lemma:

Lemma 5.2
Seien I und $\tilde{I}$ zwei permutationsäquivalente doppelgeordnete Inzidenzmatrizen einer projektiven Ebene der Ordnung n, wobei $\tilde{I} = P\,I\,Q^\top$ mit Permutationsmatrizen P und Q, für die $Pe_i = e_i$ und $Qe_i = e_i$ für $i = 1, 2$ gilt, d. h. die ersten beiden Zeilen und ersten beiden Spalten werden durch die Permutationsäquivalenz festgehalten. Dann ist $P = \mathrm{diag}(E_2, P_1, W, \hat{P}_2)$ und $Q = \mathrm{diag}(E_2, P_2, W, \hat{P}_1)$, wobei $P_1, P_2 \in \mathcal{S}_{n-1}$, $W \in \mathcal{S}_n$ und $\hat{P}_i = P_i \otimes W$ für $i = 1, 2$ sind. Die Permutationsteile von I und $\tilde{I}$ unterscheiden sich also bis auf Vertauschungen der inneren Blockspalten und Blockzeilen nur durch Konjugation der Permutationsmatrizen mit W. Insbesondere sind die Zyklenstrukturen bis auf Vertauschen der inneren Blocklinien gleich.

Beweis:
Wir wissen schon aus Lemma 5.1, daß $P = \mathrm{diag}(E_1, T_1, \hat{T}_2)$ und $Q = \mathrm{diag}(E_1, T_2, \hat{T}_1)$ sind mit Permutationsmatrizen $T_1, T_2 \in \mathcal{S}_n$, wobei $\hat{T}_i = (T_i \otimes E_n) \cdot \mathrm{diag}(W_1^i, \ldots, W_n^i)$ sind mit geeigneten $W_1^i, \ldots, W_n^i \in \mathcal{S}_n$ für $i = 1, 2$. Weiterhin gilt durch die Fixierung der zweiten Zeile und zweiten Spalte, daß $T_i e_1 = e_1$ für $i = 1, 2$. Somit ist $T_i = \mathrm{diag}(E_1, P_i)$ mit geeigneten Permutationsmatrizen $P_i \in \mathcal{S}_{n-1}$. Folglich ist auch $\hat{T}_i = \mathrm{diag}(W_1^i, \hat{P}_i)$ mit $\hat{P}_i = (P_i \otimes E_n) \cdot \mathrm{diag}(W_2^i, \ldots, W_n^i)$ und geeigneten Permutationsmatrizen $W_1^i, \ldots, W_n^i \in S_n$ für $i = 1, 2$.
Betrachten wir die Wirkung der Permutationsäquivalenz auf die 0-te Blockzeile von I, so erhalten wir durch die Rechtsmultiplikation mit $Q^\top$ bis auf die Reihenfolge eine Blockzeile der Form $(W_1^{1\top}, \ldots, W_n^{1\top})$. Nach der Linksmultiplikation mit P ändert sich die Blockzeile zu $(W_1^2\,W_1^{1\top}, \ldots, W_1^2\,W_n^{1\top})$. Damit nun $\tilde{I}$ wieder doppelgeordnet ist, muß diese Blockzeile wieder aus lauter Einheitsmatrizen bestehen, es muß also $W_1^2\,W_1^{1\top} = \cdots = W_1^2\,W_n^{1\top} = E_n$ gelten. Dies heißt jedoch gerade, daß $W_1^2 = W_1^1 = \cdots = W_n^1 =: W$. Es ist also $\hat{P}_1 = (P_1 \otimes E_n) \cdot \mathrm{diag}(W, \ldots, W) = P_1 \otimes W$. Mit analoger Überlegung an der 0-ten Blockspalte folgt $\hat{P}_2 = P_2 \otimes W$ mit demselben $W \in \mathcal{S}_n$. Insgesamt ist somit $P = \mathrm{diag}(E_2, P_1, W, \hat{P}_2)$ und $Q = \mathrm{diag}(E_2, P_2, W, \hat{P}_1)$ mit

$\hat{P}_i = P_i \otimes W$ für $i = 1, 2$. Eine Permutationsäquivalenz mit diesen beiden Matrizen bewirkt beim Permutationsteil aber bis auf Vertauschen der inneren Blocklinien gerade das Konjugieren der Permutationsmatrizen mit W. Daher bleiben insbesondere die Zyklenstrukturen der Permutationsmatrizen erhalten.

□

Dieses Lemma nutzen wir nun für unseren Test zweier Inzidenzmatrizen I_1 und I_2 auf Permutationsäquivalenz. Hierbei gehen wir schrittweise vor. Zuerst formen wir zu jeder geeigneten Kombination zweier Zeilen und Spalten die Matrix I_1 zu einer doppelgeordneten permutationsäquivalenten Inzidenzmatrix $\tilde{I}_1$ um, welche diese Zeilen und Spalten als erste Zeilen bzw. Spalten hat. Dann prüfen wir, ob sich $\tilde{I}_1$ in I_2 über eine Permutationsäquivalenz umformen läßt, welche die ersten beiden Zeilen bzw. Spalten festläßt. Da dies nach Lemma 5.2 genau dann der Fall ist, wenn bis auf Vertauschen der inneren Blocklinien die Permutationsmatrizen des Permutationsteils konjugiert zueinander sind, berechnen wir zunächst die erste Blockzeile von $\tilde{I}_1$ und prüfen, ob sie bis auf die Reihenfolge der Blöcke zu einer der Blockzeilen von I_2 konjugiert ist. Weil die Zyklenstruktur unter der Konjugation erhalten bleibt, liefert uns der Vergleich der Zyklenstrukturen der Permutationsmatrizen der Blockzeilen einen Schnelltest, durch welchen wir schon von vornherein viele Blockzeilen ausschließen können. Stimmen die Zyklenstrukturen überein, so versuchen wir, die erste Blockzeile von $\tilde{I}_1$ per Konjugation in die entsprechende Blockzeile von I_2 überzuführen. Sind wir hierbei erfolgreich, so ist die Reihenfolge der Blockspalten durch diese Transformation schon festgelegt und wir haben daher bei den Transformationen der übrigen Blockzeilen von $\tilde{I}_1$ zu den übrigen Blockzeilen von I_2 nicht mehr allzuviele Freiheiten, wodurch diese relativ schnell überprüft werden können.
Effektiver läßt sich dies noch gestalten, indem wir $\tilde{I}_1$ nicht zu jeder Kombination zweier Zeilen und Spalten berechnen, sondern nur die erste Zeile und Spalte von $\tilde{I}_1$ festlegen. Dann gibt es für die zweite Zeile und zweite Spalte je n Möglichkeiten, so daß $\tilde{I}_1$ wieder doppelgeordnet ist. Anstatt nun jedoch zu jeder dieser Möglichkeiten $\tilde{I}_1$ entsprechend umzuformen und die erste Blockzeile zu berechnen, wenden wir eine entsprechende Umformung auf I_2 an, d. h. zu jeder dieser n^2 Möglichkeiten für die zweite Zeile und zweite Spalte formen wir I_2 in eine permutationsäquivalente doppelgeordnete Inzidenzmatrix $\tilde{I}_2$ um, so daß die erste Zeile und erste Spalte von I_2 die erste Zeile bzw. erste Spalte von $\tilde{I}_2$ ist und die ausgesuchte Zeile und Spalte die zweite Zeile und zweite Spalte von $\tilde{I}_2$ bilden. Die hierdurch erhaltenen neuen Blockzeilen der $\tilde{I}_2$ speichern wir ebenfalls ab. Nun ist genau dann $\tilde{I}_1$ zu I_2 permutationsäquivalent über eine Permutationsäquivalenz, welche die erste Zeile und erste Spalte festläßt, wenn $\tilde{I}_1$ zu einer der Matrizen $\tilde{I}_2$ permutationsäquivalent ist über eine Permutationsäquivalenz, welche die ersten beiden Zeilen und Spalten festhält. Dies reduziert den benötigten Rechenaufwand erheblich, da wir nur einmal am Anfang zu jeder Kombination der

zweiten Zeile und zweiten Spalte die Inzidenzmatrizen $\tilde{I}_2$ berechnen müssen, während wir ansonsten bei jeder Kombination der ersten Zeile und ersten Spalte von $\tilde{I}_1$ diese Umformungen zu jeder der n^2 Kombinationen von zweiter Zeile und zweiter Spalte an $\tilde{I}_1$ berechnen müßten.
Kommen wir nun jedoch zum programmiertechnischen Teil, unserem Programm-Paket permAequi. Dieses ist in folgende Dateien aufgeteilt:

Main.cpp
Das Hauptprogramm. Hier wird das Programm gesteuert und die Ordnung der projektiven Ebene durch die Einbindung der entsprechenden Datei specOrdnung3.h, ..., specOrdnung9.h vereinbart.

specOrdnung3.h, ..., specOrdnung9.h
Diese Dateien enthalten die Ordnung der projektiven Ebene und die verschiedenen Zyklenstrukturen fixpunktfreier Permutationen dieser Länge.

splPerm.h
Die Klasse SplPerm, eine Klasse zur Speicherung einer Permutation ohne ihre Zyklenstruktur.

permut.h
Die Klasse Permut, eine Klasse zur Speicherung von Permutationen mit ihrer Zyklenstruktur.

permutBZ.h
Die Klasse PermutBZ, eine Klasse zur Speicherung einer Blockzeile.

getclock.h
Funktionen von F. Schmitt zur Auswertung der benötigten Prozessorzeit.

Betrachten wir zunächst die Klassen näher, welche eine geeignete Datenstruktur zur Beschreibung der Permutationsmatrizen und der Blockzeilen bereitstellen.

SplPerm
Diese Klasse ist zur Beschreibung einer Permutation, bei welcher die Zyklenstruktur nicht von Bedeutung ist. Sie ähnelt in ihrer Funktionalität stark der Klasse SimpelPerm des Programm-Pakets Inzidenz, weshalb wir nicht näher auf ihre Implementierung eingehen.

Permut
Diese Klasse beschreibt eine Permutation mit ihrer Zyklenstruktur, welche wiederum durch eine einfache Zahl vom Typ Integer codiert ist. Sie ähnelt in ihrer Funktionalität stark der Klasse Perm des Programm-Pakets Inzidenz. Daher gehen wir auch auf ihre Implementierung nicht näher ein.

PermutBZ
Die Objekte dieser Klasse beschreiben eine Blockzeile des Permutationsteils. Hierzu werden die Permutationsmatrizen der Blockzeile, mit Ausnahme des

ersten Blocks, der ja stets aus der Einheitsmatrix besteht, in dem Array bz als Permutationen der Klasse Permut mit ihrer Zyklenstruktur gespeichert. Weiterhin enthält das Array pBzO Zeiger auf die Permutationen von bz, allerdings nach der Zyklenstruktur geordnet. Diese Zyklenstrukturen sind außerdem nochmals in dem Array ordnungen nach der Größe geordnet gespeichert.

Bevor wir nun zur Implementierung des eigentlichen Algorithmus kommen, betrachten wir noch einige Hilfsfunktionen.

initM()
Diese Hilfsfunktion bringt die auf Permutationsäquivalenz zu testenden Matrizen matrix1 und matrix2 in eine für das weitere Programm geeignetere Form. Hierzu werden in M1z zu jeder Zeile von matrix1 diejenigen Spalten gespeichert, in denen matrix1 eine Eins enthält und in M1s zu jeder Spalte diejenigen Zeilen gespeichert, in denen matrix1 eine Eins enthält. Analog dazu werden auch M2z und M2s aus matrix2 erstellt.

initMatBz()
In dieser Hilfsfunktion erstellen wir zunächst matBz2. Dieses dreidimensionale Array enthält in matBz2[i][j][k] die k-te Blockzeile derjenigen doppelgeordneten Matrix, die aus matrix2 entsteht, indem wir die j-te Blockspalte mit der 0-ten Blockspalte vertauschen, die entstehende Matrix wieder doppelordnen, indem wir die Blockzeilen mit der Inversen der Permutationsmatrix in der j-ten Blockspalte von links multiplizieren, und dann analog die i-te Blockzeile mit der 0-ten Blockzeile vertauschen und die Blockspalten mit der Inversen der Permutationsmatrix in der i-ten Blockzeile von rechts multiplizieren. Somit enthält matBz2 zu jeder möglichen Kombination einer Zeile und einer Spalte die Blockzeilen einer zu matrix2 permutationsäquivalenten doppelgeordneten Matrix, welche als erste Zeile bzw. Spalte die erste Zeile bzw. Spalte von matrix2 hat und als zweite Zeile bzw. Spalte die ausgesuchte Zeile bzw. Spalte. Diese Blockspalten bringen wir noch in eine Ordnung, indem wir in bzGeordnet Zeiger auf die Blockzeilen von matBz2 speichern und diese Zeiger gemäß der lexikographischen Ordnung auf den geordneten Vektoren der Zyklenstrukturen, die ja in ordnungen gespeichert sind, ordnen. Wie bereits bei Lemma 5.2 angesprochen, ermöglicht uns dieser zunächst hohe Aufwand einen effektiven Test, ob eine gegebene doppelgeordnete Inzidenzmatrix zu matrix2 permutationsäquivalent ist über eine Permutationsäquivalenz, welche die erste Zeile und erste Spalte festläßt, indem wir überprüfen, ob die Permutationsmatrizen des Permutationsteils bis auf Vertauschen der Blockzeilen und Blockspalten zu den Blockzeilen von matBz2, welche zu einer der Umformungen von matrix2 gehören, konjugiert sind. Da unter der Konjugation die Zyklenstruktur erhalten bleibt, liefert uns der Vergleich der Zyklenstrukturen ein schnelles Verfahren, um die Konjugiertheit gewisser Blockzeilen von vornherein auszuschließen.

initBlockZeile()

In dieser Funktion erstellen wir zu einer geeigneten permutationsäquivalenten Umformung $\tilde{I}_1$ von matrix1 eine Blockzeile als Objekt der Klasse PermutBZ. Hierfür müssen wir noch einige Vorbereitungen treffen, bevor wir die Funktion aufrufen können. Zuerst legen wir im Array ersteZeilen bzw. ersteSpalten die ersten $n+1$ Zeilen bzw. Spalten von $\tilde{I}_1$ als Zeiger auf die entsprechenden Zeilen bzw. Spalten von M1z bzw. M1s fest. Damit ist der Permutationsteil bereits bis auf Konjugation der einzelnen Permutationsmatrizen bestimmt. Nun legen wir noch die relativen Spalten einer Blockspalte und damit den gesamten Permutationsteil fest. Danach speichern wir zu jeder Spalte bzw. Zeile von matrix1, welche Blockspalte, relative Spalte und relative Zeile sie in der umgeformten Matrix einnehmen in den Arrays blockSpaltenZeiger, relSpaltenZeiger bzw. relZeilenZeiger. Dabei speichern wir nicht die relativen Zeilen bzw. relativen Spalten selbst, sondern Zeiger auf das Array relative, welche die tatsächliche relative Zeilennummer bzw. relative Spaltennummer enthält. Dies vereinfacht uns später die Konjugation der ganzen Permutationsmatrizen des Permutationsteils, indem wir einfach die Zahlen von relative ändern.

Nach diesen Vorbereitungen läßt sich die erste Blockzeile von $\tilde{I}_1$ einfach berechnen, indem wir zu jeder Zeile, welche mit der dritten Spalte inzident ist, die durch deren relativen Zeilennummer gegebene Zeile der Matrix blockZeile belegen, indem wir zu jeder mit dieser Zeile inzidenten Spalte die durch deren Blockspalte gegebene Spalte von blockZeile mit dem durch deren relative Spalte gegebenen Wert belegen. Dann enthält blockZeile das durch die erste Blockzeile gegebene lateinische Quadrat und kann daraus leicht in eine Blockzeile der Klasse PermutBZ umgewandelt werden. In ähnlicher Weise läßt sich natürlich auch jede andere Blockzeile von $\tilde{I}_1$ berechnen.

isPE()

In dieser Funktion steuern wir den gesamten Test auf die Permutationsäquivalenz zweier Inzidenzmatrizen matrix1 und matrix2. Zunächst bringen wir, wie schon angesprochen, die beiden Matrizen mithilfe der Funktion initM() in eine für den weiteren Verlauf geeignetere Form. Danach setzen wir die erste Spalte und dann auch die erste Zeile von $\tilde{I}_1$ als eine mit der ersten Spalte inzidenten Zeile fest. Hierdurch ist die Inzidenzmatrix von $\tilde{I}_1$ noch keineswegs vollständig eindeutig, doch benötigen wir für den weiteren Algorithmus nur jeweils eine beliebige doppelgeordnete Umformung von matrix1, welche diese ausgesuchte Spalte bzw. Zeile als erste Spalte bzw. Zeile hat. Daher bringen wir die nächsten n Zeilen bzw. Spalten, welche durch die erste Spalte und erste Zeile schon bis auf die Reihenfolge festgelegt sind, in irgendeine Reihenfolge und speichern sie zusammen mit der erste Zeile bzw. Spalte in den Arrays ersteZeilen bzw. ersteSpalten als Zeiger auf die entsprechende Zeile bzw. Spalte von M1z bzw. M1s. Zu jeder Spalte bzw. Zeile von matrix1 berechnen wir dann blockSpaltenZeiger, relSpaltenZeiger und relZeilenZeiger,

die Blockspalte, relative Spalte und relative Zeile, welche diese bei $\tilde{I}_1$ nach der Umformung einnimmt.

Nach diesen Vorbereitungen berechnen wir in der Funktion initBlockZeile() die erste Blockzeile $\mathcal{F} = (F_1, \ldots, F_n)$ von $\tilde{I}_1$. Diese Blockzeile überprüfen wir jetzt, ob sie mit einer Blockzeile $\mathcal{G} = (G_1, \ldots, G_n)$ von matBz2 bis auf die Reihenfolge konjugiert ist, wobei die Blockzeilen nach Konstruktion zu $F_1 = E_n = G_1$ normiert sind. Als ersten Schnelltest vergleichen wir hierzu die beiden geordneten Vektoren der Zyklenstrukturen, wie sie durch ordnungen der Klasse PermutBZ gegeben sind, wodurch sich bereits viele Blockzeilen ausschließen lassen. Sind die Zyklenstrukturen gleich, so versuchen wir, $\mathcal{F}$ per Konjugation und Vertauschen der Blockspalten in $\mathcal{G}$ überzuführen. Hierzu ordnen wir zunächst die Permutationsmatrizen von $\mathcal{G}$ nach der Zyklenstruktur, was zu der Blockzeile $\mathcal{G}' = (G'_1, \ldots, G'_n)$ führt, wobei die Ordnung auf den Zyklenstrukturen geeignet sein soll, so daß $G'_1 = E_n = F_1$. Diese Ordnung der Permutationsmatrizen ist durch das Array pBzO der Klasse PermutBZ gegeben. Nun berechnen wir nacheinander zu jedem F_j, $j \geq 2$, welches dieselbe Zyklenstruktur wie G'_2 hat, ein $T \in \mathcal{S}_n$, so daß $T\,F_j\,T^{-1} = G'_2$. Mit dieser Permutation konjugieren wir dann die Blockzeile $\mathcal{F}$, speichern die entstehenden Permutationsmatrizen in konjugiertBZ und T im Array relative. Als nächstes berechnen wir den Zentralisator $C_{\mathcal{S}_n}(G'_2)$. Da dies sehr häufig geschehen muß, ist unsere schnelle Berechnung des Zentralisators mit Hilfe von Lemma 4.14 von noch entscheidenderer Bedeutung als im Programm-Paket Inzidenz. Die restlichen Permutationen probieren wir mithilfe der Funktionen vor1() und zurueck1() per Konjugation auf die Permutationen von $\mathcal{G}'$ abzubilden. Sind wir hiermit erfolgreich, so versuchen wir in rechneFertig() auch die anderen Blockzeilen der Umformung von matrix1 auf die Blockzeilen der zu $\mathcal{G}$ gehörenden Umformung überzuführen. Gelingt uns auch dies, so sind die beiden Matrizen permutationsäquivalent und wir berechnen in berechneLoesung() die zugehörige Permutationsäquivalenz. Zuletzt prüfen wir noch in testeLoesung(), ob die gefundene Lösung tatsächlich die gesuchte Permutationsäquivalenz ist.

vor1()

In dieser Funktion ist bereits in konjugiertBZ die Blockzeile $\mathcal{F}' = (F'_1, \ldots, F'_n)$ gegeben, welche aus $\mathcal{F}$ durch geeignete Konjugation entsteht, wobei $F'_{q(j)} = G'_j$ für $j < i$ gilt. Hierbei ist $q : \{1, \ldots, i-1\} \to \{1, \ldots, n\}$ eine geeignete durch das Array bsOReihenfolge gegebene Injektion. Nun versuchen wir, ein F'_j, j minimal, per Konjugation mit einer Permutation T des Zentralisators $C_{\mathcal{S}_n}(G'_1, \ldots, G'_{i-1})$ auf G'_i abzubilden. Hierbei ist es natürlich vorteilhaft, wenn der Zentralisator möglichst klein ist. Daher ordnen wir die Zyklenstrukturen wiederum nach der Größe des Zentralisators, diesmal ist jedoch, im Gegensatz zum Programm-Paket Inzidenz, diejenige Zyklenstruktur die kleinste, die den kleinsten Zentralisator hat. Hierdurch sichern wir, daß der Zentralisator zumindest für $i = 3$ minimal ist. Finden wir nun ein solches T,

so konjugieren wir die ganze Blockzeile $\mathcal{F}'$ damit und speichern die konjugierte Blockzeile wieder in konjugiertBZ. Die Permutation T multiplizieren wir mit der durch das Array relative gegebenen Permutation, und speichern das Ergebnis wiederum in dem Array, so daß relative jederzeit angibt, mit welcher Permutation die Blockzeile $\mathcal{F}$ konjugiert werden muß, um die Blockzeile in konjugiertBZ zu erhalten. Zuletzt setzen wir bsOReihenfolge[i] = j, erhöhen den Index i um eins und wiederholen diese Funktion. Finden wir dagegen kein F'_j, welches über eine Permutation des Zentralisators $C_{\mathcal{S}_n}(G'_1, \ldots, G'_{i-1})$ zu G_i konjugiert ist, so erniedrigen wir den Index i um eins und gehen in der Funktion zurueck1() einen Schritt zurück.

zurueck1()
Führten alle Versuche, die durch konjugiertBZ gegebene Blockzeile $\mathcal{F}'$ per Konjugation derart umzuformen, daß $F'_{q(j)} = G'_j$ ist für $j \leq i$, wobei wiederum q die durch bsOReihenfolge gegebene Injektion ist, zu keinem Ergebnis, so suchen wir ein neues F_j, $j > q(i)$ minimal, so daß $TF_jT^{-1} = G'_i$ mit einem $T \in C_{\mathcal{S}_n}(G'_1, \ldots, G'_{i-1})$ ist. Sind wir hierbei erfolgreich, so konjugieren wir $\mathcal{F}'$ mit T und speichern die entstehende Blockzeile in konjugiertBZ. Dann multiplizieren wir T mit der durch relative gegebene Permutation, so daß das Array wieder die richtige Permutation enthält, mit der $\mathcal{F}$ konjugiert werden muß, um die durch konjugiertBZ gegebene Blockzeile zu erhalten. Nun setzen wir noch bsOReihenfolge[i] = j, erhöhen den Index i um eins und gehen in der Funktion vor1() erneut einen Schritt vorwärts. Sind wir dagegen nicht erfolgreich, so gehen wir durch erneutes Aufrufen dieser Funktion einen weiteren Schritt zurück.

rechneFertig()
In dieser Funktion ist bereits $\mathcal{F}$ erfolgreich per Konjugation und Blockspaltenvertauschung auf $\mathcal{G}'$ abgebildet worden, woraus sich durch geeignete Blockspaltenvertauschung auch leicht eine Transformation von $\mathcal{F}$ in $\mathcal{G}$ berechnen läßt. Sei $\tilde{I}'_1$ die Matrix, welche aus $\tilde{I}_1$ entsteht, indem wir dieselbe Transformationen anwenden, welche wir genutzt haben, um $\mathcal{F}$ zu $\mathcal{G}$ umzuformen, d. h. wir konjugieren die Blöcke des Kerns von $\tilde{I}_1$ mit der durch das Array relative gegebenen Permutation und vertauschen dann entsprechend unsere Blockspalten. Die erste Blockzeile $\tilde{\mathcal{F}}'$ von $\tilde{I}'_1$ ist dabei die Umformung von $\mathcal{F}$, weshalb $\tilde{\mathcal{F}}' = \mathcal{G}$ gilt. Sei weiterhin $\tilde{I}_2$ die zu $\mathcal{G}$ gehörende Umformung der zweiten Matrix matrix2. Dann gibt es genau dann eine Permutationsäquivalenz von $\tilde{I}_1$ und $\tilde{I}_2$, welche die ersten beiden Zeilen bzw. Spalten festläßt, die Blockspalten gemäß der Umformung von $\tilde{I}_1$ zu $\tilde{I}'_1$ vertauscht und dabei die erste Blockzeile $\mathcal{F}$ von $\tilde{I}_1$ auf die Blockzeile $\mathcal{G}$ abbildet, wenn die Blockzeilen von $\tilde{I}'_2$ konjugiert sind zu den Blockzeilen von $\tilde{I}_2$ über eine Permutationsmatrix aus dem Zentralisator $C_{\mathcal{S}_n}(\tilde{F}'_1, \ldots, \tilde{F}'_n) = C_{\mathcal{S}_n}(G_1, \ldots, G_n)$. Dieser Zentralisator kann glücklicherweise nach Lemma 3.7 nicht allzu groß werden, so daß sich die restlichen Rechnungen in Grenzen halten. Mit Hilfe der Funktion vor2() berechnen wir daher jeweils die nächste Blockzeile

von $\tilde{I}_1'$, wobei die Konjugation der Permutationsmatrizen von $\tilde{I}_1$ durch unsere Konstruktion mit dem Array relative automatisch gegeben ist. Daraufhin prüfen wir diese Blockzeile, ob die Zyklenstrukturen ihrer Permutationsmatrizen mit denen einer Blockzeile von $\tilde{I}_2$ übereinstimmen. Hierbei ist auch die Reihenfolge der Permutationsmatrizen zu beachten, da sie bereits durch die bisherigen Transformationen festgelegt ist. Bei Übereinstimmung versuchen wir, die beiden Blockzeilen per Konjugation mit einer Permutationsmatrix aus dem Zentralisator $C_{\mathcal{S}_n}(G_1, \ldots, G_n)$ ineinander überzuführen. Bei Erfolg wiederholen wir dies mit der nächsten Blockzeile von $\tilde{I}_1'$, ansonsten versuchen wir mit Hilfe der Funktion zurueck2() die vorherige Blockzeile von $\tilde{I}_1'$ auf eine andere Blockzeile von $\tilde{I}_2$ zu transformieren. Erreichen wir hiermit, alle Blockzeilen von $\tilde{I}_1'$ in Blockzeilen von $\tilde{I}_2$ umzuformen, so sind die beiden Matrizen und damit auch matrix1 und matrix2 permutationsäquivalent. Die zugehörige Permutationsäquivalenz berechnen wir dann in berechneLoesung() und prüfen zur Sicherheit nochmals in testeLoesung(), ob sie wirklich die verlangten Eigenschaften erfüllt.

Als Compiler wurde wiederum der GNU-C++-Compiler unter Linux genutzt, welcher durch den Befehl „g++ -O Main.cpp“ das (optimierte) ausführbare Programm unter „a.out“ speichert. Das Programm kann auf zwei verschiedene Weisen aufgerufen werden, einmal durch „a.out datei1 datei2“, wobei datei1 und datei2 die beiden doppelgeordneten Inzidenzmatrizen der projektiven Ebenen im pbm-Format enthalten. Hierbei erkennt das Programm selbständig anhand der Formate der Matrizen, ob die Dateien die gesamte Inzidenzmatrix oder lediglich den Permutationsteil enthalten. In der Datei „ausgabe.txt“ wird dann abgelegt, ob, und wenn ja, über welche Permutationen die beiden Inzidenzmatrizen permutationsäquivalent sind. Die zweite Art des Aufrufs mittels „a.out all nummer1 anfang2 ende2“ dagegen dient dazu, eine Inzidenzmatrix mit gleich mehreren anderen Inzidenzmatrizen auf Permutationsäquivalenz zu testen, um auch die 367 gefundenen Inzidenzmatrizen für Ordnung 9 effektiv bewältigen zu können. Hierbei müssen die Dateien, welche die Inzidenzmatrizen enthalten alle die Form „matx_y.pbm“ haben, wobei x die Ordnung der projektiven Ebene und y den Index der Inzidenzmatrix angeben. Der Dateiname der ersten Inzidenzmatrix, ergibt sich dadurch, daß wir für y die durch nummer1 gegebene Zahl setzen, die Dateinamen der anderen Matrizen, welche alle mit der ersten Matrix verglichen werden sollen, ergeben sich dadurch, daß y alle Zahlen von anfang2 bis ende2 durchläuft. Die Ergebnisse der Vergleiche werden dann in 4 Dateien festgehalten, einmal in der Datei „kurzReport.txt“, welche nur die Dateinamen der zur ersten Inzidenzmatrix permutationsäquivalenten Inzidenzmatrizen enthält. Dagegen wird in „ausfuehrlich.txt“ zusätzlich noch die Permutationsäquivalenzen, über die die Matrizen permutationsäquivalent sind, gespeichert. Die Datei „ausgabe.txt“ enthält Informationen, wie die benötigte Rechenzeit, während „fehler.txt“ alle Fehler festhält, sei es daß der Test der

berechneten Permutationsäquivalenz fehlschlägt, oder daß eine Datei sich nicht einlesen läßt.

Der Vergleich ist so effektiv, daß sich ein Zwischenspeichern bei unseren berechneten Inzidenzmatrizen erübrigt. So benötigt bei Ordnung 9 ein Vergleich der mit dem Programm-Paket NeuInzidenz als erstes berechneten Inzidenzmatrix in „mat9_0.pbm“ mit den restlichen 366 Inzidenzmatrizen weniger als 10 Minuten auf einem Pentium IV-Rechner mit 2, 4 GHz. Allerdings variieren die benötigten Rechenzeiten beträchtlich mit den verglichenen Inzidenzmatrizen, so benötigt ein Vergleich der Inzidenzmatrix der desarguesschen Ebene in „mat9_312.pbm“ mit den restlichen Matrizen immerhin $7\frac{1}{2}$ Stunden auf demselben Rechner. Hier zeigt sich eine starke Abhängigkeit der Rechendauer von den Zyklenstrukturen der Permutationsmatrizen der verglichenen Inzidenzmatrizen, wobei es für den Algorithmus günstig ist, wenn die Blockzeilen Permutationsmatrizen von möglichst vielen verschiedenen Zyklenstrukturen enthalten. Doch selbst in diesem ungünstigen Fall ist die Rechenzeit akzeptabel und ein wichtiger Fortschritt gegenüber dem Programm „pe“ von B. Schwarz [23], mit welchem bereits der Test nur zweier Inzidenzmatrizen dieser Größe auf Permutationsäquivalenz sich nicht in absehbarer Zeit behandeln läßt. Allerdings ist dieser Geschwindigkeitsgewinn dadurch erkauft, daß wir uns bei dem neuen Programm auf Inzidenzmatrizen projektiver Ebenen beschränken müssen.

5.2 Komplexitätsbetrachtungen des Algorithmus

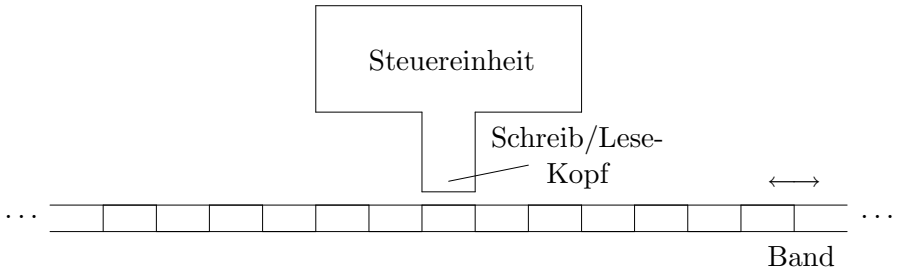
Nachdem sich unser Algorithmus als überraschend schnell erwies, stellt sich natürlich die Frage, wie schnell genau er das Problem der Permutationsäquivalenz löst. Um zu präzisieren, was „schnell“ für einen Algorithmus bedeutet, benötigen wir zunächst eine kleine Einführung in die theoretische Informatik (vgl. Wagner [26]).

Definition 5.3
Ein **Alphabet** ist eine nichtleere Menge Σ. Ihre Elemente heißen **Zeichen** oder **Buchstaben**. Ein **Wort der Länge** n über dem Alphabet Σ ist eine endliche Folge $(p_1, \ldots, p_n)$ von Zeichen $p_1, \ldots, p_n \in \Sigma$. Die Länge des Wortes x stellen wir hierbei durch $|x|$ dar. Meist läßt man die Klammern und Kommata weg und schreibt $p_1 \ldots p_n$ anstatt $(p_1, \ldots, p_n)$. Die Menge aller Wörter über dem Alphabet Σ wird mit $\Sigma^\star$ bezeichnet. Eine beliebige Teilmenge $L \subseteq \Sigma^\star$ heißt **Sprache**. Ein **Problem** über dem Alphabet Σ ist eine Funktion $c : \Sigma^\star \to \Sigma^\star$. Zu einer Sprache $L \subseteq \Sigma^\star$ ist das **Entscheidungsproblem** zu L durch die Funktion

$$c_L : \Sigma^\star \to \Sigma^\star,\ x \mapsto \begin{cases} \text{Ja} & \text{falls } x \in L \\ \text{Nein} & \text{sonst} \end{cases}$$

gegeben. In diesem Fall identifizieren wir die Sprache L mit ihrer charakteristischen Funktion c_L.

Im Falle des Entscheidungsproblems gehen wir natürlich davon aus, daß „Ja“ und „Nein“ Wörter über dem Alphabet Σ sind. Ansonsten lassen sich zur Unterscheidung auch beliebige andere Wörter verwenden. Diese Definitionen sind von der Idee geprägt, ein gegebenes Problem c durch einen Algorithmus zu lösen, d. h. wir haben eine Rechenmaschine mit einem Algorithmus-Programm gegeben, welches nach Eingabe eines Wortes $x \in \Sigma^\star$ nach endlich vielen Rechenschritten das gesuchte Wort $c(x)$ ausgibt. Hierbei müssen wir unsere Vorstellung über die verwendete Rechenmaschine genauer spezifizieren. Ein Ansatz für eine solche Rechenmaschine ist die **Turingmaschine**.



Eine Turingmaschine besteht aus einer Steuereinheit, welche endlich viele Zustände annehmen kann, einem Schreib/Lese-kopf, im folgenden einfach nur Kopf genannt, und einem unendlich langen Band, welches bezüglich des Kopfes bewegt werden kann. Hierbei kann der Kopf jeweils das Zeichen des gerade betrachteten Feldes lesen oder auch überschreiben. Die Funktionsweise der Turingmaschine ist hierbei, daß am Anfang ein Wort $x \in \Sigma^\star$, das Eingabewort, auf dem Band gespeichert ist und der Kopf auf den Anfang des Wortes positioniert ist. Die restlichen, nicht durch das Eingabewort belegten Speicherstellen des Bandes enthalten das besondere Zeichen $\Box$, das Blank-Zeichen. Weiterhin befindet sich die Steuereinheit zu Beginn in einem Startzustand. Die Wirkungsweise der Turingmaschine besteht nun darin, daß der Kopf jeweils das aktuelle Zeichen, welches sich auf der Speicherstelle unterhalb des Kopfes befindet, einliest und dann abhängig vom eingelesenen Zeichen und vom eingenommenen Zustand seinen Zustand ändert, das aktuelle Zeichen mit einem neuen Zeichen überschreibt und den Kopf auf dem Band um eine Speicherstelle nach links oder rechts bewegt oder aber stehenbleibt. Gelangt die Turingmaschine durch iterierte Anwendung dieser Übergangsvorschrift zu einem ausgezeichneten Zustand, dem Endzustand, so bricht die Maschine die Berechnung ab. Das jetzt auf dem Band stehende Wort ist das Ausgabewort. Wünschenswert ist es natürlich, wenn zu einem Problem c die Turingmaschine zu jedem Eingabewort x mit dem Ausgabewort $c(x)$ abbricht, da dann die Turingmaschine das Problem c **löst**. Eine mathematisch präzise Formulierung dieser Idee einer Rechenmaschine bietet die folgender Definition.

Definition 5.4
Eine **deterministische Turingmaschine** oder kurz **Turingmaschine** ist ein Quintupel $M = (\Sigma, Z, f, z_0, z_1)$, wobei gilt:

- Σ ist ein Alphabet mit $\square \in \Sigma$.
- Z ist eine endliche Menge mit $z_0, z_1 \in Z$, die **Zustandsmenge**.
- $f : Z \backslash \{z_1\} \times \Sigma \to Z \times \Sigma \times \{L, O, R\}$ ist eine Funktion, die **Übergangsfunktion**.
- $z_0 \in Z$ ist der Startzustand.
- $z_1 \in Z$ ist der Endzustand.

Ist hierbei f keine Funktion, sondern eine totale Relation, d. h. zu jedem $(z, \sigma) \in Z \backslash \{z_1\} \times \Sigma$ gibt es (mindestens) ein $(z', \sigma', s) \in Z \times \Sigma \times \{L, O, R\}$ mit $((z, \sigma), (z', \sigma', s)) \in f$, so ist M eine **nicht-deterministische Turingmaschine**.

Die Steuereinheit der nicht-deterministischen Turingmaschine kann mehrere Zustände auf einmal annehmen und das Band kann an einer Speicherstelle von mehreren jeweils mit den entsprechenden Zuständen korrelierende Zeichen auf einmal belegt werden. Dies stellt man sich am leichtesten so vor, daß sich die nicht-deterministische Turingmaschine in mehrere parallel arbeitende Turingmaschinen aufspalten kann, für jede durch f gegebene Zielsituation eine.

Neben dieser vorgestellten 1-Band-Turingmaschine gibt es auch mehrbandige, d. h. es gibt mehrere Speicherbänder und Köpfe, doch zeigt sich, daß sich diese durch eine einbandige Maschine simulieren lassen, wenn auch mit erhöhtem Aufwand. Daneben gibt es auch weitere Ansätze einer einen Algorithmus ausführenden Maschine, wie der Ries-, Ram- oder Mini-Riesmaschine, welche zum Teil unsere realen Computer sogar besser modellieren, doch zeigt sich, daß sich alle diese Maschinen durch eine Turingmaschine ersetzen lassen und umgekehrt auch die Turingmaschine sich durch jede dieser Maschinen simulieren läßt. Selbst durch das mächtigere Konzept der nicht-deterministischen Turingmaschine lassen sich nicht mehr Probleme lösen als mit der deterministischen Turingmaschine. Es scheint somit ein übergeordnetes Konzept der Lösbarkeit eines Problems zu geben, welches nicht von der verwendeten Maschine abhängt. Daher macht die folgende Definition Sinn.

Definition 5.5
Ein Problem c heißt **lösbar**, wenn es eine Turingmaschine gibt, die das Problem löst. Eine Sprache L heißt **entscheidbar**, wenn das Entscheidungsproblem c_L lösbar ist.

Es zeigt sich, daß nicht jedes Problem lösbar, insbesondere nicht jede Sprache entscheidbar ist. Doch auch bei den lösbaren Problemen stellt sich die Frage, wie effektiv sich diese lösen lassen. Hierbei messen wir die Effektivität eines Algorithmus mit Hilfe der Laufzeit.

Definition 5.6
Die **Laufzeitfunktion** einer Turingmaschine M ist eine Abbildung $t_M : \Sigma^\star \to \mathbb{N}$, definiert durch

$$t_M(x) := \begin{array}{l}\text{Anzahl der Schritte, welche } M \text{ bei Eingabe} \\ \text{des Wortes } x \text{ bis zum Stop ausführt.}\end{array}$$

Sei $t : \mathbb{N} \to \mathbb{N}$ eine Abbildung. Wir sagen, eine Turingmaschine M löst das Problem c in der Zeit t, wenn M das Problem c löst und $t_M(x) \leq t(n)$ für alle Wörter x der Länge n. Eine Turingmaschine M löst das Problem c **in polynomieller Zeit**, falls es ein Polynom p gibt, so daß M das Problem c in der Zeit p löst. Analog sagen wir, eine Turingmaschine M entscheidet eine Sprache L in der Zeit t, wenn M das zugehörige Entscheidungsproblem c_L in der Zeit t löst, und M entscheidet die Sprache L in polynomieller Zeit, wenn M das Entscheidungsproblem c_L in polynomieller Zeit löst.

Auf analoge Weise lassen sich auch die Laufzeiten für andere Modelle von Algorithmusmaschinen, wie den schon erwähnten Ries-, Ram- oder Mini-Ries-maschinen, aber auch für die nicht-deterministische Turingmaschine definieren. Bei der Rechenzeit ist man meist weniger an der tatsächlichen Zeit interessiert, als vielmehr am Wachstum der Rechenzeit mit der Länge des Eingabeworts. Daher benutzt man das Landausche O-Symbol, wobei eine Turingmaschine ein Problem c genau dann in der Zeit $O(t)$ löst, wenn es ein $C > 0$ gibt, so daß die Turingmaschine das Problem in der Zeit $C \cdot t + C$ löst. Konzentrieren wir uns auf Entscheidungsprobleme, so sind zwei Klassen von Sprachen von besonderen Interesse.

Definition 5.7

$$\begin{array}{lcl}\mathbf{P} & := & \{L \mid \text{Die Sprache } L \text{ ist in polynomieller Zeit durch eine} \\ & & \quad \underline{\text{deterministische}} \text{ Turingmaschine entscheidbar}\} \\ \mathbf{NP} & := & \{L \mid \text{Die Sprache } L \text{ ist in polynomieller Zeit durch eine} \\ & & \quad \underline{\text{nicht-deterministische}} \text{ Turingmaschine entscheidbar}\}\end{array}$$

Wir haben uns bei der Definition der Klasse **P** speziell auf Turingmaschinen bezogen. Wir könnten analog eine Klasse **P** bezüglich anderer Algorithmusmaschinen, wie der Ries-, Ram-, Mini-Ries- oder auch der mehrbandigen Turingmaschine definieren, doch zeigt sich, daß diese Maschinen und die deterministische Turingmaschine sich gegenseitig simulieren können, wobei der Mehraufwand durch die Simulation polynomiell beschränkt bleibt. Daher ist die Klasse **P** bezüglich jeder dieser Algorithmusmaschinen gleich.

Ebenso ist die deterministische Turingmaschine ein Spezialfall der nicht-deterministischen Turingmaschine, weshalb $\mathbf{P} \subseteq \mathbf{NP}$ gilt. Umgekehrt läßt sich auch eine nicht-deterministische Turingmaschine durch eine deterministische Turingmaschine simulieren, wobei mit den heute bekannten Methoden der Rechenaufwand eventuell exponentiell anwächst, d. h. ein Problem aus der Klasse **NP** läßt sich mit einer deterministischen Turingmaschine in exponentieller Zeit, also in der Zeit $O(e^p)$ mit einem geeigneten Polynom p, lösen. Ob dieser enorme Anstieg des Rechenaufwandes prinzipieller Natur ist oder nur von der Unzulänglichkeit der heute bekannten Methoden herrührt, ob also **NP** echt größer als **P** ist, oder beide Klassen gleich sind, ist eines der großen, ungelösten Probleme der Informatik. Relevanz für die Praxis erhält dieses doch recht akademisch anmutende Konzept einer Algorithmusmaschine dadurch, daß sich auch die grundlegenden Operationen eines heutigen Computers durch eine deterministische Turingmaschine in polynomieller Zeit durchführen lassen. Daher bilden die von einem Computer in polynomieller Zeit entscheidbaren Sprachen gerade die Klasse **P**. Wir zählen deshalb in Zukunft bei den Komplexitätsbetrachtungen die Anzahl der Schritte bei einem Computerprogramm. Eine nicht-deterministische Turingmaschine dagegen läßt sich heute nur in sehr beschränktem Maße durch große Parallelrechner umsetzen. Zusätzliche Bedeutung gewinnt dieses Problem dadurch, daß sich für viele wichtige Probleme zeigen läßt, daß sie aus der Klasse **NP** sind, die Frage, ob sie aus **P** sind, sich dagegen nur schwer oder gar nicht beantworten läßt. So ist auch das Problem der Permutationsäquivalenz aus **NP**. Um dies zu zeigen, benötigen wir jedoch noch das folgende, einfach nachprüfbare Kriterium für die Klasse **NP** (vgl. Wagner [26]).

Satz 5.8
Eine Sprache L ist genau dann in **NP**, *wenn es eine Sprache $L' \in \mathbf{P}$ und ein Polynom p gibt mit*

$$x \in L \Leftrightarrow \text{ es existiert ein Wort } y \text{ mit } |y| \leq p(|x|) \text{ und } (x, y) \in L'.$$

Unser Problem der Permutationsäquivalenz zweier $(0,1)$-Matrizen läßt sich als Entscheidungsproblem auffassen, wobei die zugehörige Sprache durch $\text{PE} := \{(M_1, M_2) \mid M_1 \text{ und } M_2 \text{ sind permutationsäquivalent}\}$ definiert ist. Dann gilt das folgende Lemma (vgl. B. Schwarz [23]).

Lemma 5.9
Die Sprache PE *gehört zur Klasse* **NP**.

Beweis:
Sei $L' := \{((M_1, M_2), (\pi_1, \pi_2)) \in (\{0,1\}^{l \times m} \times \{0,1\}^{l \times m}) \times (S_l \times S_m) \mid l, m \in \mathbb{N} \text{ und } M_2 = P_{\pi_1} \cdot M_1 \cdot P_{\pi_2}^{\top}\}$, wobei P_{π_i} die zur Permutation π_i gehörende Permutationsmatrix sei. Dann ist $L' \in \mathbf{P}$, denn wir müssen nur einerseits prüfen, ob die Eingabe sinnvoll ist, also ob M_1 und M_2 jeweils $(0,1)$-Matrizen

vom gleichen Format und π_1 und π_2 Permutationen von entsprechend passender Länge sind, wofür $O(l\,m)$ Schritte ausreichen, und andererseits die Zeilen und Spalten von M_1 gemäß π_1 und π_2 vertauschen und das Ergebnis mit M_2 vergleichen, wofür ebenfalls $O(l\,m)$ Schritte ausreichen. Somit ist L' in $O(l\,m)$ entscheidbar. Es gilt

$$x = (M_1, M_2) \in \mathrm{PE} \Leftrightarrow \text{ es existiert } y = (\pi_1, \pi_2) \text{ mit } (x, y) \in L'.$$

Um Satz 5.8 anwenden zu können, müssen wir somit nur noch nachweisen, daß $|y|$ bezüglich $|x|$ polynomiell beschränkt bleibt. Zunächst benötigen wir für eine $(0,1)$-Matrix vom Format $(l \times m)$ zumindest $l\,m$ Speicherplätze, somit ist $|x| = O(l\,m)$. Eine Permutation aus S_m läßt sich dagegen durch m Zahlen zwischen 1 und m beschreiben, also durch ein Wort der Länge $O(m\log(m))$. Damit haben die beiden Permutationen eine Länge von $O(l\log(l) + m\log(m))$ und es ist $|y| \leq O(|x|^2)$.

□

Es ist unklar, ob PE $\in$ **P** gilt, zumindest ist noch kein polynomieller Algorithmus bekannt. Ein solcher Algorithmus wäre von besonderem Interesse, läßt sich doch das in der Informatik viel bearbeitete Problem der Isomorphie zweier gegebener Graphen in polynomieller Zeit auf das Problem der Permutationsäquivalenz ihrer Inzidenzmatrizen zurückführen. Doch ist auch für dieses Teilproblem von PE bisher kein polynomieller Algorithmus bekannt (vgl. Schwarz [23] und Schöning [22]). Auch unser Algorithmus entscheidet nur eine Teilsprache von PE, nämlich PE$' := \{(M_1, M_2) \in \mathrm{PE} \mid M_1, M_2$ sind Inzidenzmatrizen projektiver Ebenen$\}$. Es läßt sich mit analogen Beweis wie zu Lemma 5.9 auch PE$' \in$ **NP** zeigen. Wie bereits angesprochen, reicht es zur Einordnung unseres Algorithmus in die Komplexitätsklassen, die Anzahl der benötigten Rechenschritte unserer Umsetzung als C++-Programm abzuschätzen. Dies hat zudem noch den Vorteil, daß wir dadurch ein besseres Bild vom tatsächlich benötigten Rechenaufwand des Programms auf einem realen Computer erhalten.

Sei im folgenden n die Ordnung der zu vergleichenden projektiven Ebenen und $N = n^2 + n + 1$ die Größe der zugehörigen Inzidenzmatrizen. Wir werden nun zeigen, daß unser Programm die Sprache PE$'$ in $O(e^{\frac{3}{2}n\ln n})$ Schritten entscheidet. Hierzu zählen wir der Kürze halber die Vorbereitungen, wie das Umformen der Matrizen in eine programmiertechnisch geeignetere Datenstruktur, nicht zur Rechenzeit hinzu, da diese Operationen nur einmal zu Beginn durchgeführt werden und auch ihre Hinzunahme die Komplexitätsklasse nicht ändert. Zunächst liefert die Festsetzung der ersten Spalte, als eine der N Spalten, und ersten Zeile, als eine der $n+1$ mit der ersten Spalte inzidenten Zeilen, einen Faktor von $O(Nn) = O(n^3)$.

Zählen wir nun die nötigen Operationen zur Berechnung der ersten Blockzeile als Objekt der Klasse PermutBZ. Die Arrays zahlen sind bereits in $O(n^2)$ Operationen belegt. Die Ermittlung der Zyklenstruktur als Variable ordnung benötigt zunächst zur Berechnung des geordneten Vektors der

Zyklenlängen $O(n^2)$ Operationen. Diesen Vektor vergleichen wir dann mit allen Zyklenstrukturen von Permutationen der Länge n. Hierfür benötigen wir $O(n\,p(n))$ Operationen, wenn $p(n)$ die Anzahl der Zyklenstrukturen ist. Dies ist gerade die Anzahl der Zahlenpartitionen von n, welche sich durch die Näherungsformel von Ramanujan zu $p(n) \approx \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{\frac{2n}{3}}} = O\left(\frac{1}{n} e^{\sqrt{\frac{2}{3}}\pi\sqrt{n}}\right)$ abschätzen läßt (vgl. Conway [8]). Zwar können wir uns auf die fixpunktfreien Zyklenstrukturen beschränken, von welchen es nur $p(n) - p(n-1)$ gibt, doch bleibt hiervon die Komplexität unberührt. Wir benötigen somit zur Berechnung einer Blockzeile als Objekt der Klasse PermutBZ insgesamt $O\left(n^2\right) + n\left(O\left(n^2\right) + O\left(n\frac{1}{n} e^{\sqrt{\frac{2}{3}}\pi\sqrt{n}}\right)\right) = O\left(n\, e^{\sqrt{\frac{2}{3}}\pi\sqrt{n}}\right)$ Operationen.

Schätzen wir nun die Anzahl der Rechenschritte ab, welche wir zur Ermittlung der verschiedenen Möglichkeiten benötigen, die erste Blockzeile auf eine der $n^2(n-1)$ Blockzeilen von matBz2 per Konjugation und Vertauschen der Blockspalten abzubilden. Zunächst vergleichen wir bei den beiden Blockzeilen die geordneten Vektoren der Zyklenstrukturen, was bereits in $O(n)$ Schritten erledigt ist. Bei Übereinstimmung berechnen wir zunächst den Zentralisator der ersten Permutation der Blockzeile. Dieser kann maximal aus $\left(\frac{n}{2}\right)!\, 2^{\frac{n}{2}}$ Permutationen bestehen, falls n gerade ist und die Permutation aus lauter 2-Zyklen besteht. Wegen der Stirling'schen Näherungsformel $n! \approx \sqrt{2\pi} e^{\left(n+\frac{1}{2}\right)\ln n - n}$ ist daher der Zentralisator in $O\left(n\left(\frac{n}{2}\right)!\, 2^{\frac{n}{2}}\right) = O\left(n e^{\left(\frac{n}{2}+\frac{1}{2}\right)\ln\frac{n}{2} - \frac{n}{2}} e^{\frac{n}{2}\ln 2}\right) = O\left(n e^{\frac{1}{2}((n+1)\ln n - n)}\right)$ Rechenschritten bestimmt (vgl. Bronstein, Semendjajew [5]). Bei der Konjugation der Blockzeile gibt es maximal $(n-1)!$ Möglichkeiten, wie die Reihenfolge der Permutationsmatrizen, die Einheitsmatrix ausgenommen, variieren kann. Im ungünstigsten Fall konjugieren wir bei jeder dieser veränderten Reihenfolgen n mal die n Permutationsmatrizen mit allen Permutationen des Zentralisators, wobei die Konjugation einer Permutation jeweils $O(n)$ Rechenschritte benötigt. Daher ist zusammen die Anzahl der Operationen zur Berechnung der verschiedenen Möglichkeiten, wie die erste Blockzeile auf eine der Blockzeilen von matBz2 per Konjugation und Vertauschen der Blockspalten umgeformt werden kann, durch $n^2(n-1)\, O\left(n e^{\frac{1}{2}((n+1)\ln n - n)}\right) O((n-1)!\, n^3) = O\left(n^3\, n\, e^{\frac{1}{2}((n+1)\ln n - n)} n^2 e^{\left(n+\frac{1}{2}\right)\ln n - n}\right) = O\left(n^7\, e^{\frac{3}{2}n\ln n - \frac{3}{2}n}\right)$ nach oben beschränkt, wobei wiederum $n!$ durch die Stirling'sche Formel abgeschätzt ist.

Berechnen wir nun die Anzahl der Operationen, die notwendig sind, um die übrigen $n-2$ Blockzeilen per Konjugation mit einer Permutation aus Z, dem Zentralisator der Permutationsmatrizen der ersten Blockzeile, auf die entsprechenden Blockzeilen von matBz2 abzubilden. Nach Lemma 3.7 ist $|Z| \leq n$. Daher müssen wir im ungünstigsten Fall jede der $n-2$ Blockzeilen mit den höchstens n Permutationen von Z konjugieren und probieren, ob sie dann einer der entsprechenden $n-2$ Blockzeilen von matBz2 gleicht, wobei

die Reihenfolge der Blockspalten bereits durch die erste Blockzeile festgelegt ist. Somit läßt sich dieser Teil des Algorithmus bereits in $O(n^7)$ Operationen bewältigen.

Zusammen ergibt sich somit für den Algorithmus eine Komplexität von $O\left(n^3 \cdot n\, e^{\sqrt{\frac{2}{3}}\pi\sqrt{n}} \cdot n^7\, e^{\frac{3}{2}n\ln n - \frac{3}{2}n} \cdot n^7\right) = O\left(n^{18} e^{\frac{3}{2}n\ln n - \frac{3}{2}n + \sqrt{\frac{2}{3}}\pi\sqrt{n}}\right)$. Wegen $\lim\limits_{n\to\infty} n^{18} e^{\sqrt{\frac{3}{2}}\pi\sqrt{n} - \frac{3}{2}n} = 0$ können wir dies durch $O\left(e^{\frac{3}{2}n\ln n}\right)$ nach oben abschätzen. Unser Programm-Paket permAequi liefert somit keinen polynomiellen Algorithmus.

Das Programm-Paket pe von B. Schwarz [23] muß dagegen im schlimmsten Fall alle $N \cdot (n+1)! \cdot (n!)^2$ Permutationsäquivalenzen, welche eine gegebene doppelgeordnete Inzidenzmatrix einer projektiven Ebene wieder in eine doppelgeordnete Inzidenzmatrix überführt, auf eine Inzidenzmatrix anwenden und dann mit der anderen Matrix vergleichen. Dies liefert eine Komplexität von $O\left(N^2 \cdot N\,(n+1)!\,(n!)^2\right) = O\left(n^7\, e^{3\left(n+\frac{1}{2}\right)\ln n - 3n}\right)$, wobei wir wiederum die Stirling'sche Näherungsformel verwendet haben. Dies ist fast das Quadrat der Komplexität unseres verbesserten Algorithmus, wodurch sich die Effektivität des Programms permAequi auch auf theoretischer Ebene dokumentieren läßt. Ein weiterer interessanter Spezialfall ist der Ausschluß der Permutationsäquivalenz zweier Inzidenzmatrizen. Hierfür müssen mit dem Programm pe selbst im besten Fall $N\,(n+1)!\,n!$ Permutationsäquivalenzen betrachtet werden, womit eine Komplexität von $O\left(N^2 \cdot N\,(n+1)!\,n!\right) = O\left(n^8\, e^{2n\ln n - 2n}\right)$ gegeben ist (siehe Schwarz [23]). Unser verbesserter Algorithmus schließt somit im ungünstigsten Fall schneller die Permutationsäquivalenz zweier Matrizen aus, als das Programm pe im günstigsten Fall.

Weiterhin scheint unser Algorithmus in der Praxis meist effektiver zu arbeiten als diese „worst case"-Betrachtungen vermuten lassen, tritt doch der kritische Fall, daß alle Permutationsmatrizen dieselbe Zyklenstruktur haben, gerade bei den desarguesschen Ebenen auf, welche eine besonders große Kollineationsgruppe haben. Große Kollineationsgruppen haben jedoch nach Lemma 2.7 gleich zwei günstige Auswirkungen für die Beantwortung der Frage nach der Permutationsäquivalenz. Einerseits gibt es dann nur wenige Inzidenzmatrizen dieser Ebene. Hierbei ist zu erwarten, daß sich diese Eigenschaft auf die doppelgeordneten Inzidenzmatrizen überträgt, zumal sich dies mit der Anzahl der durch unser Programm NeuInzidenz gefundenen Matrizen deckt. Andererseits gibt es zu zwei solchen Inzidenzmatrizen gleich eine Vielzahl von Permutationsäquivalenzen, so daß recht schnell eine solche zu finden sein dürfte. Insgesamt haben wir somit durch die Beschränkung auf Inzidenzmatrizen von projektiven Ebenen und unter Ausnutzung der speziellen Anordnung durch die Doppelordnung einen wesentlich schnelleren Algorithmus als den bisherigen erstellt, der das Problem der Permutationsäquivalenz für unsere Matrizen in akzeptabler Zeit bewältigt, wenngleich auch er dies nicht in polynomieller Zeit lösen kann.

6 Die gefundenen Inzidenzmatrizen

Im folgenden wollen wir uns mit den Resultaten der ganzen Bemühungen beschäftigen, nämlich den durch das Programm-Paket NeuInzidenz gefundenen Inzidenzmatrizen. Hierbei beschränken wir uns auf jeweils eine doppelgeordnete Inzidenzmatrix aus jeder Permutationsäquivalenzklasse und stellen deren Permutationsteil im pbm-Format dar, bei dem die Einsen durch ein schwarzes Quadrat und die Nullen durch ein weißes Quadrat kenntlich gemacht werden. Wegen programmiertechnischen Schwierigkeiten erstellt unser Programm die lateinischen Quadrate bis Ordnung 2 nicht fehlerfrei, so daß mit dem Programm-Paket Inzidenz erst projektive Ebenen ab Ordnung 3 und mit dem Programm-Paket NeuInzidenz projektive Ebenen erst ab Ordnung 4 korrekt berechnet werden, ein sicherlich verschmerzbarer Schönheitsfehler angesichts der Erfolge bei der Berechnung der Ebenen höherer Ordnung. Aus Gründen der Vollständigkeit ist unsere Liste jedoch noch um die Permutationsteile der projektiven Ebenen der Ordnung 2 und 3 ergänzt. Zur besseren Überprüfbarkeit sind weiterhin die Namen der zugehörigen Dateien angegeben, wie sie vom Programm-Paket NeuInzidenz vergeben werden.

Ordnung 2:

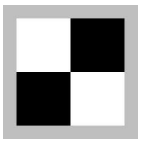


Abbildung 2:

Ordnung 3:

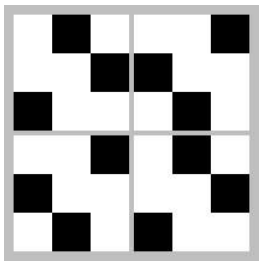


Abbildung 3: mat3_0.pbm

Ordnung 4:

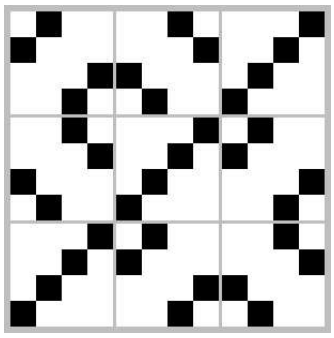


Abbildung 4: mat4_0.pbm

Ordnung 5:

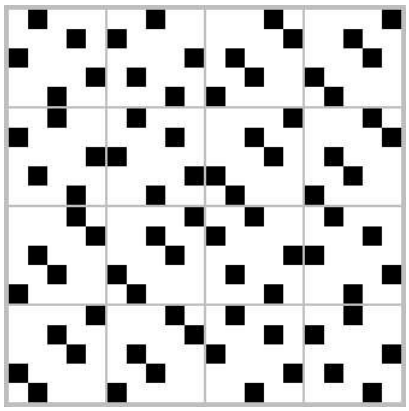


Abbildung 5: mat5_0.pbm

Ordnung 6:

Eine projektive Ebene der Ordnung 6 existiert nach Satz 2.5 (Bruck-Ryser) nicht.

Ordnung 7:

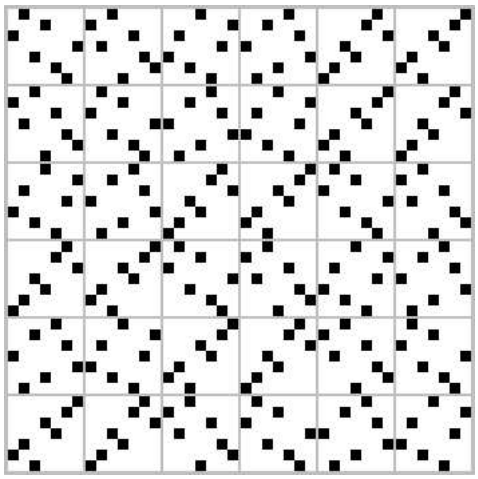


Abbildung 6: mat7_0.pbm

Ordnung 8:

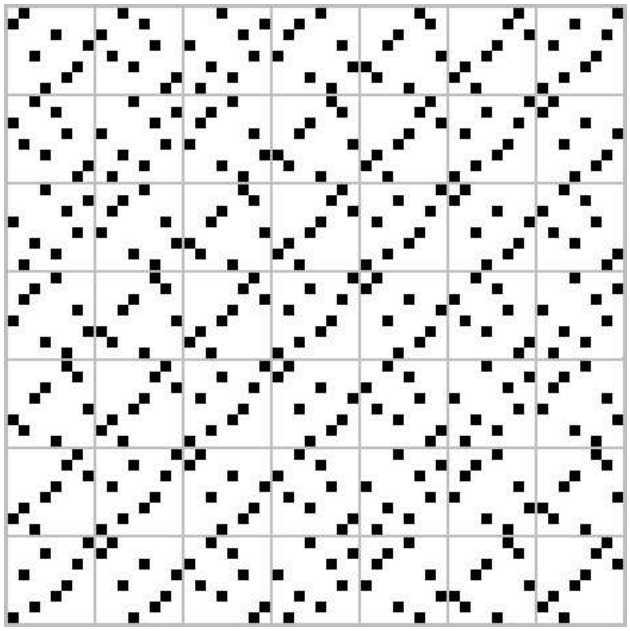


Abbildung 7: mat8_0.pbm

Bis zu Ordnung 8 gibt es jeweils nur die desarguesschen Ebenen. Auffällig an den Permutationsteilen ist, daß die Permutationsmatri-

zen der ersten Blockzeile jeweils in veränderter Reihenfolge die übrigen Blockzeilen ergeben. Dies wollen wir am Beispiel der Ordnung 8 etwas genauer untersuchen. Wir definieren die Matrizen $A, \ldots, H \in \mathcal{S}_8$ durch A sei die Einheitsmatrix, $B := (12)(35)(48)(67)$, $C := (13)(25)(46)(78)$, $D := (14)(36)(28)(57)$, $E := (15)(23)(47)(68)$, $F := (16)(34)(27)(58)$, $G := (17)(26)(45)(38)$ und $H := (18)(24)(37)(56)$, wobei wir die Permutationsmatrizen durch die zugehörigen Permutationen in der üblichen Zyklenschreibweise ausdrücken. Dann bildet $(A, \ldots, H)$ die erste Blockzeile der doppelgeordneten Inzidenzmatrix, welche den Permutationsteil aus Abbildung 7 enthält, und der Permutationsteil hat die Form

$$\begin{array}{ccccccc} B & C & D & E & F & G & H \\ C & D & E & F & G & H & B \\ D & E & F & G & H & B & C \\ E & F & G & H & B & C & D \\ F & G & H & B & C & D & E \\ G & H & B & C & D & E & F \\ H & B & C & D & E & F & G \end{array}$$

Die Permutationsmatrizen der Blockzeilen sind zyklisch verschoben. Die reduzierten lateinischen Quadrate bilden daher eine Cayley-Tafel der zyklischen Gruppe $\mathbb{Z}_7$. Bei den gefundenen Inzidenzmatrizen niedrigerer Ordnung ist dies nicht mehr so offensichtlich, doch zeigt eine genauere Untersuchung, daß auch hier die reduzierten lateinischen Quadrate isomorph zur Cayley-Tafel der zyklischen Gruppe sind. Betrachtet man die Permutationsmatrizen der Blockzeile $(A, \ldots, H)$, so fällt zunächst auf, daß mit Ausnahme der Einheitsmatrix A alle Permutationsmatrizen dieselbe Zyklenstruktur haben, sie alle aus 2-2-2-2-Zyklen bestehen. Weiterhin bilden die Permutationsmatrizen der Blockzeilen $A, \ldots, H$ eine Gruppe, die isomorph zu $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$ ist, der elementar-abelschen Gruppe der Ordnung 8. Diese Eigenschaft läßt sich auch bei den gefundenen Matrizen von Ebenen geringerer Ordnung erkennen, d. h. für die gefundenen Inzidenzmatrizen einer projektiven Ebene der Ordnung n bilden die Permutationsmatrizen einer Blockzeile eine zu der elementar-abelschen Gruppe der Ordnung n isomorphe Gruppe und es sind die Zyklenstrukturen, und insbesondere auch die Ordnungen, der Permutationsmatrizen des Permutationsteils gleich.

Ordnung 9:

Bei Ordnung 9 gibt es nun erstmals mehrere nicht isomorphe Typen von projektiven Ebenen, insbesondere auch nicht desarguessche. Unsere gefundenen Inzidenzmatrizen lassen sich mithilfe des Programm-Pakets permAequi in 4 Klassen einteilen. Zur Identifizierung der zugehörigen projektiven Ebenen verwendeten wir als Referenzmatrizen

die von Knoflíček [13] angegebenen Inzidenzmatrizen. Hiermit konnten wir die Inzidenzmatrizen folgenden projektiven Ebenen zuweisen:

Die Rechts-Fastkörper-Ebene

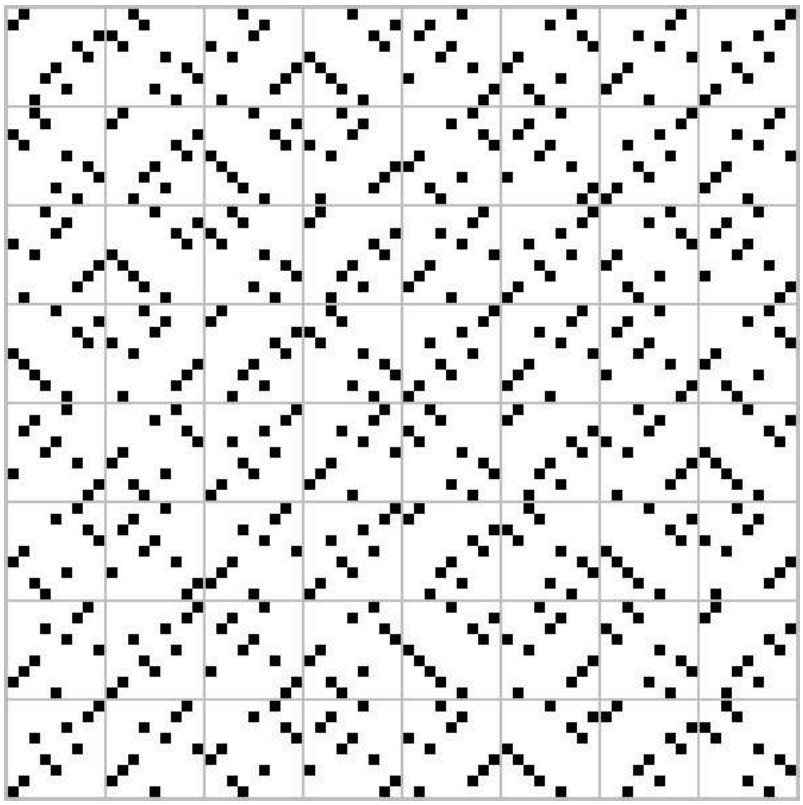


Abbildung 8: mat9_28.pbm

Auch in dieser Inzidenzmatrix zeigt sich, daß die Permutationsmatrizen der ersten Blockzeile in jeder anderen Blockzeile in veränderter Reihenfolge wiederholt werden. Weiterhin haben die Permutationsmatrizen des Permutationsteils alle dieselbe Zyklenstruktur, jedoch bestehen sie nicht aus 3-3-3-Zyklen, wie zu erwarten, sondern aus 2-2-2-3-Zyklen. Insbesondere bilden die Permutationsmatrizen einer Blockzeile keine Gruppe. Eine nähere Untersuchung der reduzierten lateinischen Quadrate zeigt, daß diese wiederum eine Cayley-Tafel einer Gruppe bilden, allerdings nicht eine Cayley-Tafel der zyklischen Gruppe $\mathbb{Z}_8$, wie nach den bisherigen Ergebnissen zu vermuten, sondern eine Cayley-Tafel der Quaternionengruppe H_8.

Die Links-Fastkörper-Ebene:

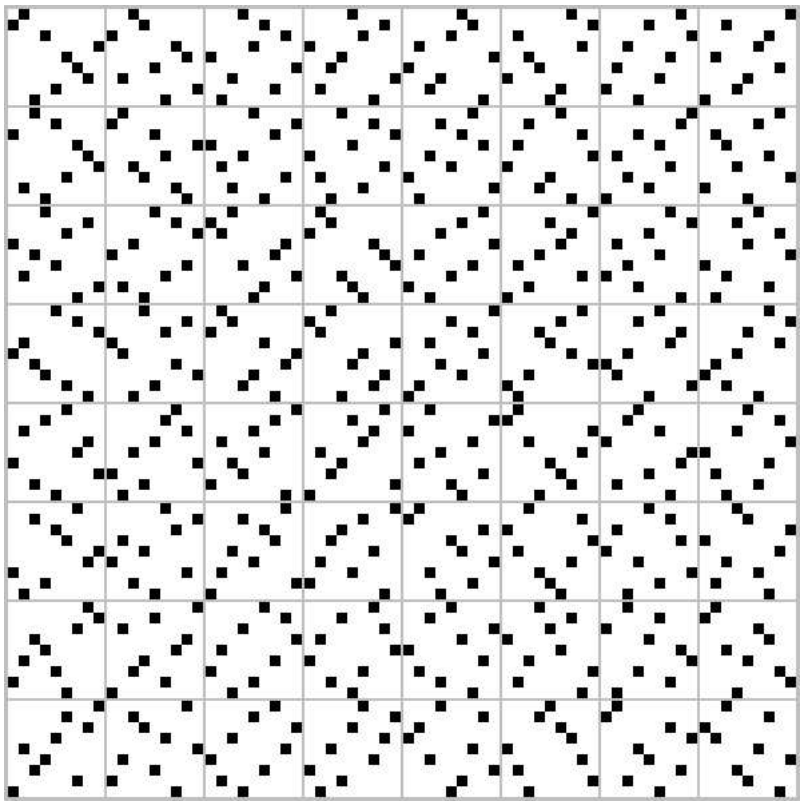


Abbildung 9: mat9_26.pbm

Dieser Permutationsteil unterscheidet sich wesentlich von den bisher betrachteten, denn differierten die Blockzeilen bisher nur in der Reihenfolge der Permutationsmatrizen, so ist hier keine solche Eigenschaft mehr zu erkennen. Eine genauere Betrachtung zeigt sogar, daß überhaupt keine der Permutationsmatrizen mehrfach vorkommt. Dies erscheint besonders überraschend, da die Links-Fastkörper-Ebene dual zur Rechts-Fastkörper-Ebene ist. Daher ist die Transponierte der Matrix aus Abbildung 8 ebenfalls der Permutationsteil einer doppelgeordneten Inzidenzmatrix der Links-Fastkörper-Ebene und läßt sich daher per Permutationsäquivalenz in unsere Matrix aus Abbildung 9 umformen, obwohl die Permutationsteile keine Gemeinsamkeiten vermuten lassen.

Die Hughes-Ebene:

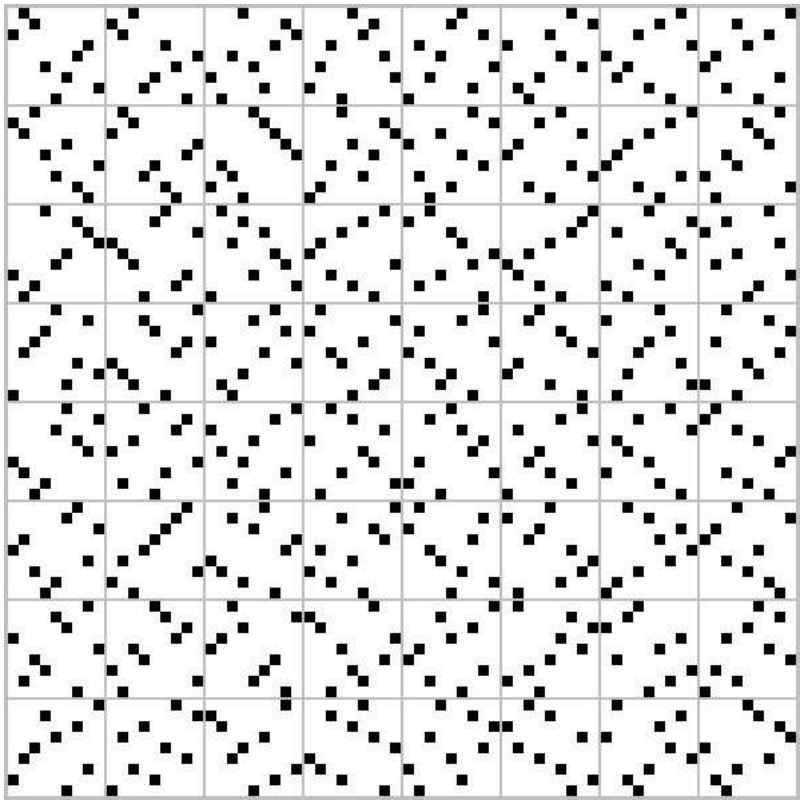


Abbildung 10: mat9_26.pbm

Auch bei diesem Permutationsteil einer Inzidenzmatrix der Hughes-Ebene läßt sich keine besondere Regel in Bezug auf die Permutationsmatrizen erkennen, denn wie beim Permutationsteil der Links-Fastkörper-Ebene von Abbildung 9 kommt auch bei diesem Permutationsteil der Hughes-Ebene keine der Permutationsmatrizen doppelt vor. Eine eingehendere Analyse der Zyklenstrukturen zeigt weiterhin, daß jede fixpunktfreie Zyklenstruktur durch eine Permutationsmatrix des Permutationsteils einer geeigneten doppelgeordneten Inzidenzmatrix der Hughes-Ebene angenommen wird. Dies steht ganz im Gegensatz zu den Inzidenzmatrizen der projektiven Ebenen kleinerer Ordnung, bei denen sich auch durch Permutationsäquivalenzen, welche die Doppelordnung erhalten, die Zyklenstruktur der Permutationsmatrizen des Permutationsteils nicht verändern läßt. Untersuchungen bei den Fastkörper-Ebenen zeigen, daß auch hier mehrere Zyklenstrukturen angenommen werden, jedoch nur die folgenden:
9, 2-3-4, 2-2-5, 2-2-2-3 und 3-3-3.

Die Desarguessche Ebene:

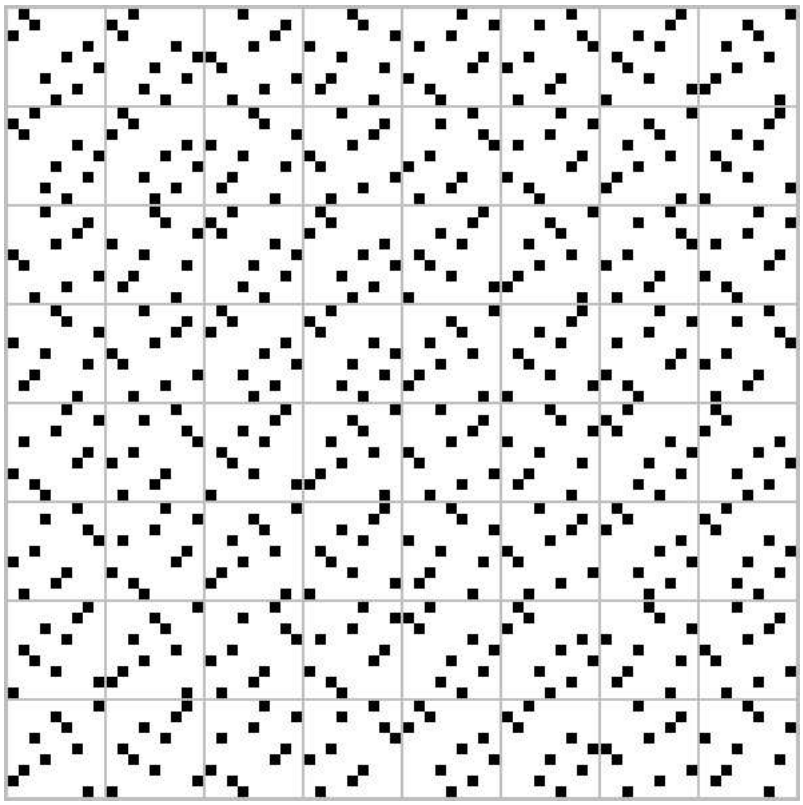


Abbildung 11: mat9_312.pbm

Bei diesem Permutationsteil unterscheiden sich die Blockzeilen wiederum nur in der Reihenfolge der Permutationsmatrizen. Auch bestätigen sich die Vermutungen aus den Beispielen der projektiven Ebenen niedriger Ordnung, so bilden die Permutationsmatrizen einer Blockzeile eine zur elementar-abelschen Gruppe $\mathbb{Z}_3 \times \mathbb{Z}_3$ isomorphe Gruppe. Die Permutationsmatrizen des Permutationsteils bestehen alle aus 3-3-3-Zyklen und ihre Zyklenstruktur läßt sich auch durch Permutationsäquivalenzen, welche die Doppelordnung erhalten nicht verändern. Weiterhin sind die reduzierten lateinischen Quadrate isomorph zur Cayley-Tafel der zyklischen Gruppe $\mathbb{Z}_8$.

Als Resümee dieses Kapitels ist mit Rechnerunterstützung für die Ordnungen $2, 3, 4, 5, 7$ und 8 gezeigt, daß es jeweils nur die desarguessche Ebene gibt, und, daß es keine projektive Ebene der Ordnung 6 gibt, im Einklang mit Satz 2.5 (Bruck-Ryser). Zudem haben wir für Ordnung 9 eine Bestätigung des ebenfalls rechnerunterstützten Ergebnisses von Lam [15], daß es

nur die 4 bekannten projektiven Ebenen gibt. Die Anzahl der berechneten Inzidenzmatrizen beträgt bei der Rechts-Fastkörper-Ebene 117, bei der Links-Fastkörper-Ebene 116, bei der Hughes-Ebene 132 und bei der desarguesschen Ebene 2. Dies steht im Einklang mit Lemma 2.7, wonach es gerade zu der desarguesschen Ebene mit ihrer großen Kollineationsgruppe verhältnismäßig wenig Inzidenzmatrizen gibt, und dies scheint sich hiernach auch auf die Anzahl der durch unser Programm berechneten Inzidenzmatrizen niederzuschlagen. Etwas verwunderlich ist auf den ersten Blick, daß sich die Anzahl der berechneten Inzidenzmatrizen bei den Fastkörper-Ebenen unterscheiden, sind die beiden doch dual zueinander, also die Inzidenzmatrizen bis auf Permutationsäquivalenz transponiert zueinander. Daher haben die beiden Ebenen auch genauso viele doppelgeordnete Inzidenzmatrizen. Durch die Reduzierung der ersten reduzierten lateinischen Quadrate auf möglichst wenige Repräsentanten je Isomorphieklasse werden jedoch jeweils Inzidenzmatrizen zusammengefaßt, wobei es, wie in diesem Fall, durchaus vorkommen kann, daß von der einen Ebene diese Inzidenzmatrizen stärker zusammengefaßt werden, also weniger Matrizen berechnet werden, als von der dazu dualen Ebene.

Wir wollen uns an dieser Stelle Gedanken über ein Problem machen, welches immer bei rechnerunterstützten Beweisen vorhanden ist, nämlich die Frage der Korrektheit der Ergebnisse, denn schließlich ist es nicht möglich, alle durchgeführten Rechnungen per Hand nachzuprüfen. Befassen wir uns daher mit der Möglichkeit, daß es eine fünfte, nicht gefundene projektive Ebene der Ordnung 9 gibt. Ihre Kollineationsgruppe wäre dann auf jeden Fall kleiner als die der desarguesschen Ebene, vermutlich sogar wesentlich kleiner, sind doch gerade die Ebenen mit großer Kollineationsgruppe noch recht gut von theoretischer Seite her zugängig. Eine kleine Kollineationsgruppe geht jedoch nach Lemma 2.7 mit einer großen Zahl an zugehörigen Inzidenzmatrizen einher. Die berechneten Beispiele lassen zudem vermuten, daß sich dies auf die durch unseren Algorithmus gefundenen Inzidenzmatrizen überträgt, zumindest bei korrekter Rechnung. Eine genauere Analyse zeigt weiterhin, daß bei den gefundenen Inzidenzmatrizen der nichtdesarguesschen Ebenen eine Vielzahl von ganz unterschiedlichen, nicht isomorphen reduzierten lateinischen Quadraten auftreten. Unter der Annahme, daß dies auch für die fragliche fünfte Ebene gilt, wäre es daher äußerst unwahrscheinlich, daß bei der Vorgabe aller dieser reduzierter lateinischer Quadrate ein Fehler auftritt, so daß die Ebene nicht gefunden würde. Daher können wir mit ziemlicher Sicherheit eine weitere, nicht gefundene Ebene ausschließen.

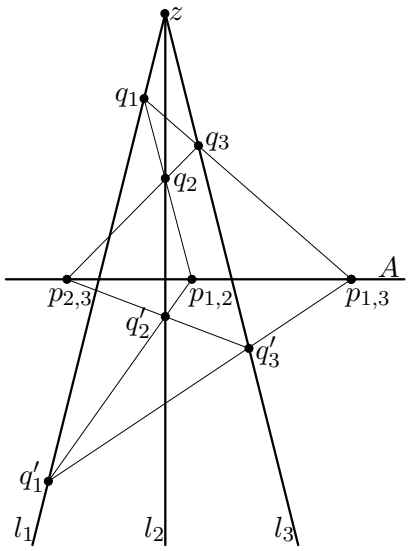
Die Untersuchungen an den Permutationsteilen der gefundenen Inzidenzmatrizen, insbesondere der Hughes-Ebene, geben keinen Hinweis auf eine Regel, nach der bestimmte Zyklenstrukturen nicht vorkommen können. Somit können wir anscheinend keine der fixpunktfreien Zyklenstrukturen als niemals in einem Permutationsteil vorkommend von vornherein ausschlie-

ßen, ohne dabei zu riskieren, damit auch eine projektive Ebene von der Berechnung auszuschließen, ein sicherlich ernüchterndes Ergebnis gerade in Hinblick auf eine mögliche Beschleunigung des Algorithmus. Beschränken wir uns dagegen auf desarguessche Ebenen, so zeigt sich durch die bisherigen Beispiele eine besondere Regel bezüglich der lateinischen Quadrate der Blockzeilen und der reduzierten lateinischen Quadrate. Es scheint somit eine eingehendere Befassung mit den desarguesschen Ebenen angebracht, weshalb wir ihnen auch das nächste Kapitel widmen.

7 Desarguessche Ebenen

An den Beispielen des letzten Kapitels sahen wir die desarguesschen Ebenen durch eine besondere Struktur des Permutationsteils auffallen. Daher wollen wir uns in diesem Kapitel überlegen, wie sich die Eigenschaft desarguessch von einer projektiven Ebene auf ihre doppelgeordnete Inzidenzmatrix überträgt. Hierzu zunächst die korrekte Definition der desarguesschen Ebenen (vgl. Stevenson [24]).

Definition 7.1 (Satz von Desargues)
Sei z ein Punkt und A eine Gerade einer projektiven Ebene $(\mathcal{P}, \mathcal{L}, \in)$. Zu drei Geraden l_1, l_2, l_3, welche sich in z schneiden, seien die Punkte $q_i, q_i' \in l_i$ für $i = 1, 2, 3$ gegeben. Hierdurch sind für $i, j = 1, 2, 3$, $i < j$ die drei Punkte $p_{i,j} := (q_i \vee q_j) \wedge (q_i' \vee q_j')$ festgelegt. Eine projektive Ebene heißt (z, A)**-desarguessch**, wenn für alle Geraden l_i durch z und alle Punkte $q_i, q_i' \in l_i$ gilt: Sind zwei der Punkte $p_{i,j}$ auf A, so auch der dritte. In diesem Fall nennen wir auch die zugehörige Inzidenzmatrix (z', A')-desarguessch, wobei z' die zu z gehörige Spalte und A' die zu A gehörige Zeile sind. Eine projektive Ebene heißt **desarguessch**, wenn sie (z, A)-desarguessch ist für alle Punkte z und alle Geraden A. Analog bezeichnen wir dann auch die zugehörige Inzidenzmatrix als desarguessch.



Es lassen sich zwei Fälle bei den (z, A)-desarguesschen projektiven Ebenen unterscheiden, insbesondere in Hinblick auf eine Berechnung der Ebenen, nämlich der Fall, daß z auf A liegt, und der andere Fall, daß z nicht auf A liegt. Hierauf werden wir jedoch an späterer Stelle noch genauer eingehen.

Weiterhin sind die desarguesschen projektiven Ebenen genau diejenigen, die man durch unsere Konstruktion in Beispiel 2.4 aus Vektorräumen erhält (vgl. Pickert [20]). Somit wissen wir bereits, daß es genau zu jeder Primzahlpotenz genau eine desarguessche projektive Ebene dieser Ordnung gibt. Desarguessche Ebenen zeichnen sich durch eine starke Symmetrie und eine große Kollineationsgruppe aus. Eine besondere Klasse von Kollineationen sind die axialen Kollineationen (vgl. Baer [1], p–L-transformation).

Definition 7.2
Eine Kollineation (α, β) einer projektiven Ebene heißt **axiale Kollineation** (auch (z, A)-Kollineation), falls es einen Punkt z und eine Gerade A gibt, so daß $\alpha(x) = x$ für alle Punkte x auf A und $\beta(l) = l$ für alle Geraden l durch z gilt. Man nennt dann z das **Zentrum** und A die **Achse** von (α, β). Ist $z \in A$, so spricht man von einer **Translation**, ist $z \notin A$, so spricht man von einer **Streckung**.

Ist l eine Gerade durch z und sind $a, b \in l \setminus (\{z\} \cup A)$ zwei Punkte, so gibt es höchstens eine axiale Kollineation mit Achse A und Zentrum z, die a auf b abbildet. Hierbei gehen wir der Kürze wegen im gesamten Kapitel davon aus, daß die Inzidenzrelation durch $\in$ gegeben ist, wir identifizieren also die Gerade l mit der Menge der auf ihr liegenden Punkte $[l]$. Interessant ist der Fall, wenn es zu je zwei solchen Punkten a, b eine axiale Kollineation gibt (vgl. Baer [1] Theorem 2.1, Theorem 3.1 und Theorem 6.2).

Definition 7.3
Sei z ein Punkt und A eine Gerade einer projektiven Ebene. Sei weiterhin $l \neq A$ eine Gerade durch z. Die projektive Ebene heißt (z, A)**-transitiv**, wenn es zu je zwei Punkten $a, b \in l \setminus (\{z\} \cup A)$ eine axiale Kollineation (α, β) mit Achse A und Zentrum z gibt mit $\alpha(a) = b$. In diesem Fall gibt es zu jeder Geraden $l \neq A$ durch z und $a, b \in l \setminus (\{z\} \cup A)$ eine (z, A)-Kollineation (α, β) mit $\alpha(a) = b$.

Die Eigenschaft der (z, A)-Transitivität einer projektiven Ebene bedeutet somit gerade, daß für eine (und damit auch jede) Gerade $l \neq A$, welche durch z geht, die Gruppe der (z, A)-Kollineationen (scharf) transitiv auf den Punkten $l \setminus (\{z\} \cup A)$ wirkt. Eine wichtige Verbindung zwischen den Eigenschaften (z, A)-desarguessch und (z, A)-transitiv bei projektiven Ebenen liefert der folgende Satz (vgl. Baer [1] Theorem 6.2).

Satz 7.4 (Baer)
Eine projektive Ebene ist genau dann (z, A)-desarguessch, wenn sie (z, A)-transitiv ist.

Man beachte, für die Gültigkeit dieses Satzes ist es irrelevant, ob der Fall $z \in A$ oder der Fall $z \notin A$ auftritt. Im weiteren Verlauf ist dieser Satz für

unsere Betrachtungen essentiell, denn er verknüpft die geometrische Eigenschaft des Satzes von Desargues mit der für unsere Inzidenzmatrix besser behandelbaren Existenz gewisser Kollineationen. Wählen wir hierbei spezielle Punkte und Geraden als Zentrum bzw. Achse, so können wir bereits einige Aussagen über die Form der doppelgeordneten Inzidenzmatrix treffen. Davor wollen wir uns jedoch noch kurz mit speziellen lateinischen Quadraten beschäftigen, nämlich den Cayley-Tafeln von Gruppen, wobei wir für die lateinischen Quadrate die Schreibweise aus Definition 4.6 verwenden.

Lemma 7.5
Ist das lateinische Quadrat $A = (f_1, \ldots, f_n)$ eine Cayley-Tafel einer Gruppe G, so bildet $\{f_1, \ldots, f_n\}$ eine zu G isomorphe Gruppe. Ist umgekehrt $G = \{f_1, \ldots, f_n\}$ eine überdeckungsfreie Untergruppe von S_n, so ist $A = (f_1, \ldots, f_n)$ ein zur Cayley-Tafel von G isomorphes lateinisches Quadrat.

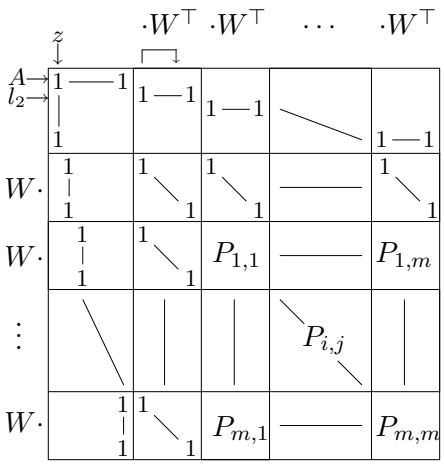
Beweis:
Sei $(G, \cdot)$ eine Gruppe über den Zahlen $\{1, \ldots, n\}$. Die zugehörige Cayley-Tafel $A = (f_1, \ldots, f_n)$ ist definiert durch $f_j(i) = i \cdot j$ für $i, j = 1, \ldots, n$. Dann ist $\varphi : G \to \{f_1, \ldots, f_n\}$, $i \mapsto f_{i^{-1}}$ ein Isomorphismus, denn es gilt die Bijektivität nach Konstruktion und es ist $\varphi(i \cdot j)(k) = f_{(i \cdot j)^{-1}}(k) = k \cdot (i \cdot j)^{-1} = k \cdot j^{-1} \cdot i^{-1} = f_{i^{-1}}(k \cdot j^{-1}) = (f_{i^{-1}} \circ f_{j^{-1}})(k) = (\varphi(i) \circ \varphi(j))(k)$ für alle $i, j, k \in G$.

Sei nun umgekehrt $A = (f_1, \ldots, f_n)$ ein lateinisches Quadrat, wobei $G = \{f_1, \ldots, f_n\}$ eine Gruppe bildet. Sei o. B. d. A. das Quadrat A normiert bezüglich erster Zeile und Spalte, d. h. sei $f_1 = \mathrm{id}$ und sei $f_i(1) = i$ für alle $i = 1, \ldots, n$. Wir definieren die Gruppe $(G', \cdot)$ auf den Zahlen $\{1, \ldots, n\}$ durch $i \cdot j = k \Leftrightarrow f_j \circ f_i = f_k$. Dann ist G' isomorph zu G per Isomorphismus $i \mapsto f_i^{-1}$ und $A = (a_{i,j})$ ist die Cayley-Tafel von G', denn für alle $i, j, k = 1, \ldots, n$ ist $a_{i,j} = k \Leftrightarrow f_j(i) = k \Leftrightarrow (f_j \circ f_i)(1) = f_j(f_i(1)) = f_k(1) \overset{*}{\Leftrightarrow} f_j \circ f_i = f_k \Leftrightarrow i \cdot j = k$ ($*$ Für die Richtung $\Rightarrow$ benötigen wir, daß die $f_k \in G$ überdeckungsfrei sind, dann gilt nämlich $f_k(1) = f_{k'}(1) \Rightarrow f_k = f_{k'}$). Somit ist A die Cayley-Tafel von G' und damit isomorph zur Cayley-Tafel von G. □

Dieses Lemma liefert uns ein einfaches Kriterium, um zu erkennen, ob ein lateinisches Quadrat $A = (f_1, \ldots, f_n)$ bis auf Isomorphie die Cayley-Tafel einer Gruppe ist. Wir müssen A nur zu $f_1 = \mathrm{id}$ normieren und dann überprüfen, ob $G := \{f_1, \ldots, f_n\}$ eine Gruppe bildet. In diesem Fall ist zudem noch A bis auf Isomorphie bereits durch den Isomorphietyp von G bestimmt, d. h. ist $\{f'_1, \ldots, f'_n\}$ eine andere Repräsentation von G in S_n, so daß $A' = (f'_1, \ldots, f'_n)$ ebenfalls ein lateinisches Quadrat ergibt, so ist A' isomorph zu A. Betrachten wir nun die Auswirkungen spezieller (z, A)-Kollineationen auf die Inzidenzmatrix.

Lemma 7.6
Seien I eine doppelgeordnete Inzidenzmatrix einer projektiven Ebene der Ordnung n und G die Gruppe ihrer (z, A)-Kollineationen, wobei z die erste Spalte und A die erste Zeile ist. Dann ist G isomorph zum Zentralisator $Z = C_{\mathcal{S}_n}(\{P_{i,j} \mid i, j = 1, \ldots, n-1\})$, wobei $P_{i,j}$ die Permutationsmatrizen des Permutationsteils sind. Weiterhin ist $|G|$ ein Teiler von n, also insbesondere $|G| \leq n$, wobei die Gleichheit genau dann eintritt, wenn die Ebene (z, A)-desarguessch ist.

Beweis:
Wir betrachten die (z, A)-Kollineationen von G als Permutationsäquivalenzen (P, Q) mit $PIQ^\top = I$ und verwenden für die Linien von I die Bezeichnungen aus Kapitel 2. Wir wollen zunächst die Form der (z, A)-Kollineationen näher bestimmen. Sei hierzu $(P, Q) \in G$. Nun werden von (P, Q) alle mit dem Zentrum $z = p_1$ inzidenten Zeilen und alle mit der Achse $A = l_1$ inzidenten Spalten festgehalten, also alle Zeilen der Rand-Blockzeile und alle Spalten der Rand-Blockspalte. Insbesondere werden also die ersten beiden Zeilen und ersten beiden Spalten fixiert und nach Lemma 5.2 sind daher $P = \mathrm{diag}(E_2, P_1, W, \hat{P}_2)$ und $Q = \mathrm{diag}(E_2, P_2, W, \hat{P}_1)$ mit geeigneten $P_1, P_2 \in \mathcal{S}_{n-1}$, $W \in \mathcal{S}_n$ und $\hat{P}_i = P_i \otimes W$ für $i = 1, 2$. Da auch die restlichen Linien der Rand-Blocklinien festgehalten werden, muß ferner $P_1 = E_{n-1} = P_2$ gelten, also $P = \mathrm{diag}(E_{n+1}, W, \ldots, W) = Q$ mit geeignetem $W \in \mathcal{S}_n$. Die Kollineationen aus G konjugieren also nur die Blöcke des Kerns mit einer Permutationsmatrix W. Damit unter dieser Wirkung die Inzidenzmatrix I erhalten bleibt, muß W aus dem Zentralisator $Z := C_{\mathcal{S}_n}(\{P_{i,j} \mid i, j = 1, \ldots, n-1\})$ sein, wobei die $P_{i,j}$ die Permutationsmatrizen des Permutationsteils sind. Umgekehrt liefert auch jede Permutationsmatrix des Zentralisators Z eine (z, A)-Kollineation aus G. Daher ist $\varphi : Z \to G$, $W \mapsto (\mathrm{diag}(E_{n+1}, W, \ldots, W), \mathrm{diag}(E_{n+1}, W, \ldots, W))$ ein Isomorphismus und Z und G sind isomorph.
Nun ist Z als Untergruppe von $C_{\mathcal{S}_n}(P_{1,1}, \ldots, P_{1,n-1})$ nach Lemma 3.7 eine Gruppe von überdeckungsfreien Permutationsmatrizen. Nach Korollar 3.6 ist dann jedoch $|Z|$ und damit auch $|G|$ ein Teiler von n, insbesondere also $|Z| = |G| \leq n$. Die Gleichheit tritt genau dann ein, wenn Z ein flächendeckendes Ensemble bildet, also Z nach Lemma 2.12 scharf transitiv auf den Einheitsvektoren $e_1, \ldots, e_n$ wirkt. Dies wiederum ist äquivalent dazu,



daß $\varphi(Z) = G$ transitiv auf den Spalten der 0-ten Blockspalte wirkt, den Punkten von $l_2 \setminus (\{z\} \cup A)$ (in diesem Fall ist $l_2 \setminus (\{z\} \cup A) = l_2 \setminus \{z\}$, da $l_2 \cap A = \{z\}$). Dies heißt jedoch gerade, daß die projektive Ebene (z, A)-transitiv, nach Satz 7.4 also (z, A)-desarguessch ist.

□

Mit diesem Lemma haben wir ein einfaches Kriterium, um zu entscheiden, ob die doppelgeordnete Inzidenzmatrix I bezüglich der erste Zeile und ersten Spalte (z, A)-desarguessch ist, indem wir den Zentralisator Z der Permutationsmatrizen berechnen, denn I ist genau dann (z, A)-desarguessch, wenn Z maximal ist, also $|Z| = n$. In diesem Fall können wir die Struktur von I noch genauer bestimmen.

Lemma 7.7
Sei I eine doppelgeordnete Inzidenzmatrix einer projektiven Ebene der Ordnung n. Sei I weiterhin (z, A)-desarguessch, wobei z die erste Spalte und A die erste Zeile von I seien. Dann gilt:

i) *die reduzierten lateinischen Quadrate unterscheiden sich nur durch Umnummerierung.*
ii) *Die inneren Blocklinien differieren nur in der Reihenfolge ihrer Permutationsmatrizen.*
iii) *Sind $P_{i,j}$ die Permutationsmatrizen des Permutationsteils, so ist der Zentralisator $Z := C_{\mathcal{S}_n}(\{P_{i,j} \mid i, j = 1, \ldots, n-1\})$ ein flächendeckendes Ensemble. Insbesondere ist $|Z| = n$.*
iv) *Ist G die Menge aller Permutationsmatrizen einer inneren Blocklinie, so gilt $G = C_{\mathcal{S}_n}(Z)$. Insbesondere ist G eine Gruppe.*

Beweis:

Sei Z wie im Lemma und sei $G = \{E, P_{1,1}, \ldots, P_{1,n-1}\}$ die Menge der Permutationsmatrizen der ersten Blockzeile. Es gilt $Z \subseteq C_{\mathcal{S}_n}(G)$, woraus mit Lemma 3.7 dann $|Z| \leq |C_{\mathcal{S}_n}(G)| \leq n$ folgt. Andererseits ist I nach Voraussetzung (z, A)-desarguessch, nach Lemma 7.6 ist also $|Z| = n$, woraus aus Ordnungsgründen die Gleichheit $Z = C_{\mathcal{S}_n}(G)$ folgt. Somit ist Z nach Lemma 3.7 ein flächendeckendes Ensemble und iii) ist gezeigt. Weiterhin ist nach Korollar 3.8 dann $G = C_{\mathcal{S}_n}(Z)$ und G eine Gruppe. Nun müssen jedoch alle Permutationsmatrizen $P_{i,j}$, $i, j = 1, \ldots, n-1$ in $C_{\mathcal{S}_n}(Z) = G$ liegen. Wegen der Überdeckungsfreiheit enthält daher jede innere Blocklinie gerade die Permutationsmatrizen von G, sie bilden also jeweils die Gruppe $G = C_{\mathcal{S}_n}(Z)$ und differieren nur in der Reihenfolge der Blöcke, womit ii) und vi) gezeigt ist.

Anders formuliert bedeutet dies: Enthält ein Block des Permutationsteils, welcher im ersten Zeilenabschnitt durch den Einheitsvektor $e_i^\top$ belegt ist, im j-ten Zeilenabschnitt den Vektor $e_k^\top$, so enthält jeder Block, der im

ersten Zeilenabschnitt durch $e_i^\top$ belegt ist, im j-ten Zeilenabschnitt $e_k^\top$. Das heißt jedoch gerade, daß die zugehörigen reduzierten lateinischen Quadrate sich nur in der Nummerierung unterscheiden, es gilt also i).

□

Nach diesem Lemma ist die Struktur der betrachteten Inzidenzmatrizen bereits stark festgelegt. So werden bei den inneren Blockzeilen die Permutationsmatrizen der ersten Blockzeile nur in veränderter Reihenfolge wiederholt und diese Permutationsmatrizen bilden eine Gruppe, d. h. nach Lemma 7.5 sind die zugehörigen lateinischen Quadrate isomorph zur Cayley-Tafel dieser Gruppe. Dies läßt sich für einen besonders schnellen Algorithmus zur Berechnung von (z, A)-desarguesschen projektiven Ebenen nutzen, bei denen das Zentrum z auf der Achse A liegt.

Hierzu nehmen wir die doppelgeordnete Inzidenzmatrix so angeordnet an, daß die erste Zeile die Achse A und die erste Spalte das Zentrum z bilden. Bei den niedrigen Ordnungen sind alle Isomorphietypen von Gruppen bekannt, so daß wir nacheinander zu jeder dieser Isomorphietypen eine Cayley-Tafel dieser Gruppe berechnen und die zugehörigen Permutationsmatrizen in die erste Blockzeile einsetzen können. Ist hierbei der Zentralisator dieser Permutationsmatrizen in $\mathcal{S}_n$ nicht maximal, so können wir diese Blockzeile auch sogleich wieder verwerfen. Neben dieser starken Einschränkung bei der ersten Blockzeile läßt sich zudem auch die Vervollständigung der Inzidenzmatrix erheblich beschleunigen. Da sich die übrigen inneren Blockzeilen nur in der Reihenfolge der Permutationsmatrizen unterscheiden, können wir beim Backtracking-Verfahren jeweils eine ganze Permutationsmatrix statt nur einen Zeilenabschnitt einsetzen. Zudem wirkt sich die Gruppeneigenschaft der Permutationsmatrizen positiv auf die Überprüfung der Rechtecksregel aus, denn die Permutationsmatrizen $P_{i,j}, P_{i',j}, P_{i',j'}$ und $P_{i,j'}$ erfüllen nach Satz 3.9 genau dann die Rechtecksregel, wenn $\tilde{P} := P_{i,j} P_{i',j}^{-1} P_{i',j'}$ und $P_{i,j'}$ überdeckungsfrei sind. Da die Permutationsmatrizen jedoch eine Gruppe bilden, ist auch $\tilde{P}$ ein Element dieser Gruppe. Nun sind zwei Elemente dieser Gruppe genau dann überdeckungsfrei, wenn sie verschieden sind, weshalb wir lediglich prüfen müssen, ob $\tilde{P}$ und $P_{i,j'}$ verschieden sind.

Für nicht (z, A)-desarguessche Ebenen sind durch Lemma 7.6 weitere, wenn auch im Vergleich zu den (z, A)-desarguesschen Ebenen weniger effektive Ideen gegeben, indem wir zunächst Z vorgeben. Hierbei ist Z eine Gruppe, deren Ordnung ein Teiler von n ist, so daß die Möglichkeiten für Z recht beschränkt sind. Nun sind die Blöcke des Permutationsteils alle im Zentralisator $C_{\mathcal{S}_n}(Z)$, wodurch sich bei nicht trivialem Z eine erhebliche Einschränkung gegenüber allen Matrizen von $\mathcal{S}_n$ ergibt. Für den Fall $Z = \{\mathrm{id}\}$ läßt sich jedoch auch durch diese Idee keine Beschleunigung erreichen.

Lemma 7.8
Sei I eine doppelgeordnete Inzidenzmatrix einer (z, A)-desarguesschen projektiven Ebene der Ordnung n, wobei $z = p_{n+1+1}$ die erste Spalte der 0-ten Blockspalte und $A = l_1$ die erste Zeile von I sind. Dann gilt:

i) *Das erste reduzierte lateinische Quadrat $A_1 = (f_1, \ldots, f_{n-1})$ ist isomorph zur Cayley-Tafel der Gruppe der (z, A)-Kollineationen.*

ii) *Die übrigen reduzierten lateinische Quadrate $A_2, \ldots, A_n$ sind gleich bis auf Umnummerierung und Spaltenvertauschung, also insbesondere isomorph zueinander.*

iii) *Je zwei Permutationsmatrizen des Permutationsteils derselben Blockzeile sind konjugiert zueinander, haben also insbesondere dieselbe Zyklenstruktur und dieselbe Ordnung. Genauer gilt:*
Wir nehmen A_1 als normiert bezüglich der ersten Spalte an. Fassen wir dann $f_1, \ldots, f_{n-1}$ als Permutationen über den Zahlen $2, \ldots, n$ auf und erweitern sie zu Permutationen $\hat{f}_1, \ldots, \hat{f}_{n-1}$ über den Zahlen $1, \ldots, n$ mit Fixpunkt 1, so gibt es zu je zwei Permutationen $P_{i,j}$ und $P_{i,j'}$ des Permutationsteils genau ein $\hat{f}_k$, so daß $\hat{F}_k P_{i,j} \hat{F}_k^\top = P_{i,j'}$, wobei $\hat{F}_k$ die zu $\hat{f}_k$ gehörende Permutationsmatrix, definiert durch $\hat{f}_k(i) = j \Leftrightarrow \hat{F}_k e_i = e_j$, ist. Hierbei hängt k nicht von der Blockzeile i ab.

Beweis:

Berechnen wir zunächst die (z, A)-Kollineationen, aufgefaßt als Permutationsäquivalenzen (P, Q) mit $PIQ^\top = I$. Wegen der Achse $A = l_1$ werden die ersten beiden Spalten p_1 und p_2 und die erste Zeile $A = l_1$ festgehalten. Da die zweite Zeile l_2 mit dem Zentrum $z = p_{n+1+1}$ inzidiert, wird auch diese fixiert, und nach Lemma 5.2 hat die Permutationsäquivalenz die Form $P = \mathrm{diag}(E_2, P_1, W, \hat{P}_2)$ und $Q = \mathrm{diag}(E_2, P_2, W, \hat{P}_1)$ mit geeigneten $P_1, P_2 \in \mathcal{S}_{n-1}$, $W \in \mathcal{S}_n$ und $\hat{P}_i = P_i \otimes W$ für $i = 1, 2$. Wegen der Achse $A = l_1$ müssen weiterhin die Spalten $p_3, \ldots, p_{n+1}$ fixiert werden, es ist also $P_2 = E_{n-1}$. Da außerdem das Zentrum $z = p_{n+1+1}$ festgehalten wird, ist e_1 ein Fixpunkt von W. Es ist somit $P = P(W) := \mathrm{diag}(E_2, P_1, W, \ldots, W)$ und $Q = Q(W) := \mathrm{diag}(E_{n+1}, W, \hat{P}_1)$. Die (z, A)-Kollineation konjugiert also nur die Permutationsmatrizen mit einer Matrix $W \in \mathcal{S}_n$, wobei e_1 von W fixiert wird, und vertauscht dann noch die inneren Blockspalten geeignet. Insbesondere ist die (z, A)-Kollineation schon durch die Vorgabe von W eindeutig bestimmt und die Gruppe G der (z, A)-Kollineationen ist isomorph zu der Gruppe $G' := \{W \mid (P(W), Q(W)) \text{ ist } (z, A)\text{-Kollineation}\}$.

Wir wollen nun G' näher bestimmen. Die Zeile l_{n+1+1}, die erste Zeile der 0-ten Blockzeile, geht durch das Zentrum z und die Punkte $a = p_{n+1+n+1}$, die erste Spalte der ersten Blockspalte, und $b = p_{n+1+jn+1}$, die erste Spalte der j-ten Blockspalte, liegen in $l_{n+1+1} \setminus (\{z\} \cup A)$ für $j = 1, \ldots, n-1$. Daher gibt es nach Lemma 7.4 eine (z, A)-Kollineation, welche a auf b abbildet, d. h. konjugieren wir die erste Blockspalte mit einem geeignetem $W \in G'$, so

erhalten wir die j-te Blockspalte. Dies zeigt bereits, daß zwei Permutationsmatrizen des Permutationsteils in derselben Blockzeile konjugiert zueinander sind und daher dieselbe Zyklenstruktur und Ordnung haben.

Bleibt noch W genauer zu bestimmen. Hierzu nehmen wir das erste lateinische Quadrat $A_1 = (f_1, \ldots, f_{n-1})$ zu $f_1 = \mathrm{id}$ normiert an. Da e_1 von W fixiert wird, wird das erste reduzierte lateinische Quadrat durch die Multiplikation der Permutationsmatrizen mit W von links nicht verändert. Durch die Multiplikation der Permutationsmatrizen mit $W^\top$ von rechts wird dann die erste Spalte f_1 von A_1 derart umnummeriert, daß wir die j-te Spalte f_j erhalten. Betrachten wir f_j als Permutation über den Zahlen $2, \ldots, n$ und erweitern sie zu einer Permutation $\hat{f}_j$ über den Zahlen $1, \ldots, n$ mit Fixpunkt 1, so ist gerade $W = \hat{F}_j$ die zugehörige Permutationsmatrix. Somit ist $G' = \{\hat{F}_j \mid f_j \text{ ist Spalte von } A_1\}$. Damit gilt einerseits, daß es zu zwei Permutationsmatrizen $P_{i,j}$ und $P_{i,j'}$ des Permutationsteils genau eine Spalte f_k von A_1 gibt, so daß $\hat{F}_k P_{i,j} \hat{F}_k^\top = P_{i,j'}$, wobei k nicht von der Blockzeile i abhängt, es gilt also iii). Andererseits ist $\{f_1, \ldots, f_{n-1}\}$ isomorph zu G', also auch zu G, und nach Lemma 7.5 ist A_1 isomorph zur Cayley-Tafel von G, es gilt also i).

Nun wirkt $G' = \{\hat{F}_1, \ldots, \hat{F}_{n-1}\}$ (scharf) transitiv auf $\{e_2, \ldots, e_n\}$, es gibt also für alle $j = 2, \ldots, n$ ein $\hat{F}_j \in G'$ mit $\hat{F}_j e_2 = e_j$. Bei der Inzidenzmatrix I werden durch die Multiplikation mit $P(\hat{F}_j)$ von links die relativen zweiten Zeilen auf die relativen j-ten Zeilen abgebildet, also das zweite reduzierte lateinische Quadrat A_2 an die Stelle des j-ten reduzierten lateinischen Quadrates A_j. Multiplizieren wir die Inzidenzmatrix dann noch mit $Q(\hat{F}_j)^\top$ von rechts, so werden bei den reduzierten lateinischen Quadraten noch die Spalten vertauscht und Nummerierung permutiert. Da $(P(\hat{F}_j), Q(\hat{F}_j))$ eine Kollineation ist, unterscheiden sich daher A_2 und A_j nur durch Spaltenvertauschung und Umnummerierung für alle $j = 2, \ldots, n$, es gilt also ii).

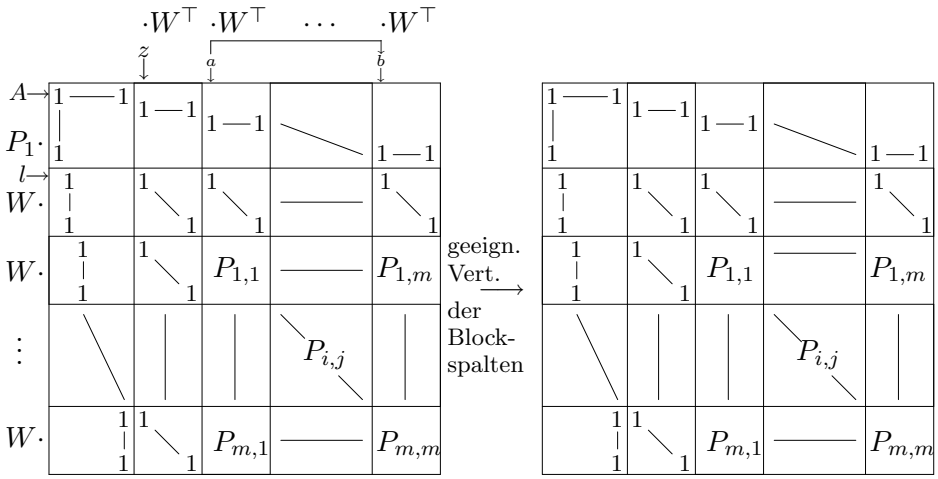


Abbildung 12: eine (z, A)-Kollineation

□

Während wir Lemma 7.7 zur Beschleunigung der Berechnung von (z, A)-desarguesschen Ebenen verwenden konnten, bei denen das Zentrum z auf der Achse A liegt, können wir dieses Lemma zur schnelleren Berechnung einer (z, A)-desarguesschen Ebene nutzen, bei der das Zentrum z nicht auf der Achse A liegt. Hierzu nehmen wir die zugehörige doppelgeordnete Inzidenzmatrix so angeordnet an, daß die Achse die erste Zeile und das Zentrum die erste Spalte der 0-ten Blockspalte bilden. Nach diesem Lemma ist dann das erste reduzierte lateinische Quadrat isomorph zur Cayley-Tafel einer Gruppe. Dies ist eine wesentliche Einschränkung, gibt es doch erheblich weniger Gruppen der Ordnung $n-1$ als Isomorphieklassen von lateinischen Quadraten.

Mit den bisherigen Lemmata lassen sich zudem bereits einige Eigenschaften der doppelgeordneten Inzidenzmatrizen von desarguesschen projektiven Ebenen erschließen. Dies wollen wir im folgenden Korollar zusammenfassen.

Korollar 7.9
Sei I eine doppelgeordnete Inzidenzmatrix der desarguesschen projektiven Ebene der Ordnung n. Dann gilt:

i) *Die Blocklinien differieren nur in der Reihenfolge ihrer Permutationsmatrizen.*
ii) *Die Permutationsmatrizen einer Blocklinie bilden eine Gruppe.*
iii) *Die reduzierten lateinischen Quadrate unterscheiden sich nur in der Nummerierung und sind isomorph zur Cayley-Tafel einer Gruppe.*
iv) *$n = p^r$ für eine Primzahl p.*
v) *die Permutationsmatrizen des Permutationsteils haben gleiche Zyklenstruktur und gleiche Ordnung. Genauer gilt, die Ordnung ist jeweils p und die Zyklen haben jeweils Länge p.*

Beweis:
i) und ii) gelten direkt nach Lemma 7.7 ii) und iv). Zudem folgt iii) aus Lemma 7.7 i) und Lemma 7.8 i). Sei nun $\mathcal{G}$ die Menge der Permutationsmatrizen der ersten Blockzeile, welche nach ii) eine Gruppe bildet. Mit Ausnahme der Einheitsmatrix haben dann nach Lemma 7.8 iii) alle Gruppenelemente gleiche Ordnung. Nun gibt es jedoch nach dem 1. Sylowsatz zu jedem Primteiler p von $|\mathcal{G}| = n$ ein Element der Ordnung p (vgl. Jacobson [11]). Daher darf n nur einen Primteiler haben, es gilt also iv): $n = p^r$ mit einer Primzahl p. Zudem hat jedes $F \in \mathcal{G}$, $F \neq E_n$ Ordnung p und die Zyklen von F haben nach Korollar 3.6 alle gleiche Länge, welche aufgrund der Ordnung p betragen muß. Nach i) überträgt sich dies dann auf alle Permutationsmatrizen des Permutationsteils, es gilt also v).

□

Die Eigenschaft iv) ist natürlich eine direkte Folgerung aus dem Struktursatz (vgl. Pickert [20]), wonach die desarguesschen Ebenen gerade die gemäß

Beispiel 2.4 von Körpern konstruierten Ebenen sind. Sie läßt sich jedoch, wie hier gesehen, bereits durch die schwächeren Voraussetzungen von Lemma 7.7 und Lemma 7.8 zeigen.

Für eine noch weitergehendere Charakterisierung der doppelgeordneten Inzidenzmatrizen der desarguesschen projektiven Ebenen wollen wir zunächst eine solche Matrix konstruieren. Hierbei orientieren wir uns an das Konzept von Knoflíček [13], welches jedoch nicht mit einer durchgängigen Beweisführung untermauert ist, so daß wir dies an dieser Stelle für die desarguesschen Ebenen nachholen. Dafür benötigen wir zunächst eine andere Geometrie, die affine Ebene (vgl. Lingenberg [17]).

Definition 7.10
Eine **affine Ebene** ist eine Inzidenzstruktur mit folgenden Eigenschaften:

i) zu je zwei verschiedenen Punkten gibt es genau eine Verbindungsgerade.

ii) zu je einem Punkt p und einer Geraden l gibt es genau eine zu l parallele Gerade l' durch p. Hierbei heißen zwei Geraden parallel, wenn sie gleich sind oder sich nicht schneiden.

iii) es gibt ein Dreieck, d. h. drei nicht kollineare Punkte.

Ist $\mathbb{K}$ ein Körper, $\mathcal{P} := \mathbb{K}^2$ und $\mathcal{L} := \{v + w\mathbb{K} \mid v, w \in \mathbb{K}^2,\ w \neq 0\}$, so bildet $A(\mathbb{K}) := (\mathcal{P}, \mathcal{L}, \in)$ eine affine Ebene, die **desarguessche affine Ebene** über dem Körper $\mathbb{K}$. Hierbei sind zwei Geraden $v + w\mathbb{K}$ und $v' + w'\mathbb{K}$ genau dann parallel, wenn die Unterräume $w\mathbb{K}$ und $w'\mathbb{K}$ gleich sind, also w und w' linear abhängig. Aus einer affinen Ebene läßt sich stets durch das sogenannte **Vervollständigen** eine projektive Ebene konstruieren, indem man jede Parallelenklasse von Geraden sich jeweils in einem ,,unendlich fernen" Punkt schneiden läßt und diese zusätzlichen Punkte noch durch eine neue, ,,unendlich ferne" Gerade verbindet. Umgekehrt erhält man durch **Schlitzen** einer projektiven Ebene eine affine Ebene, indem man von der projektiven Ebene eine Gerade und alle mit dieser Gerade inzidenten Punkte wegläßt. Insbesondere werden die desarguesschen projektiven und desarguesschen affinen Ebenen durch Vervollständigen und Schlitzen ineinander übergeführt (vgl. Lingenberg [17]). Übertragen wir das Schlitzen auf die Inzidenzmatrizen, so müssen wir von einer Inzidenzmatrix einer projektiven Ebene eine Zeile und alle mit dieser Zeile inzidenten Spalten weglassen und erhalten dann eine Inzidenzmatrix einer affinen Ebene. Schlitzen wir eine doppelgeordnete Inzidenzmatrix einer projektiven Ebene an der ersten Zeile, so erhalten wir die Inzidenzmatrix der zugehörigen affinen Ebene durch Entfernen der ersten Zeile und der Rand-Blockspalte. Die Parallelenklassen von Geraden dieser affinen Ebene sind hierbei gerade die Blockzeilen (vgl. Abbildung 13).

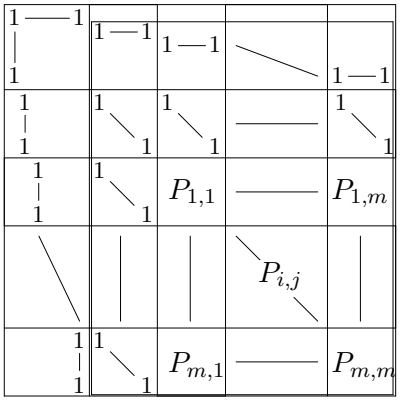


Abbildung 13: Inzidenzmatrizen projektiver und affiner Ebenen

Wir wollen nun die Inzidenzmatrix der affinen Ebene wie in Abbildung 13 angeordnet als Teilmatrix der Inzidenzmatrix der projektiven Ebene annehmen. Wir definieren auf den Linien der Inzidenzmatrix der projektiven Ebene neue Koordinaten durch $p_{(i,j)}$ sei die Spalte in der i-ten Blockspalte mit relativen Spaltenindex j und dual dazu $l_{(x,y)}$ sei die Zeile in der x-ten Blockzeile mit relativen Zeilenindex y. Hierbei wollen wir bei der Nummerierung des relativen Linienindex, wie schon bei der Nummerierung der Blocklinien mit 0 beginnen, so daß $i, j, x, y \in \{0, \ldots, n-1\}$. Die Rand-Linien, welche nicht durch diese Koordinatisierung erfaßt werden, behalten ihre alte Bezeichnung.

Fassen wir nun das Galoisfeld $\mathbb{F}_n$ als Körper über den Zahlen $\{0, \ldots, n-1\}$ auf, so erhalten wir dadurch eine natürliche Identifizierung der Spalten des Kerns mit den Vektoren von $\mathbb{F}_n^2$. Dann ist die Zeile l_{y+2} der Geraden $\begin{pmatrix} y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathbb{F}_n$ für alle $y = 0, \ldots, n-1$ zugeordnet. Die restlichen Zeilen $l_{(x,y)}$ ordnen wir nun den restlichen Geraden $\begin{pmatrix} 0 \\ y \end{pmatrix} + \begin{pmatrix} 1 \\ x \end{pmatrix} \mathbb{F}_n$ für $x, y = 0, \ldots, n-1$ zu. Für den Kern gilt hiermit: $l_{(x,y)}$ hat genau dann in der i-ten Blockspalte den Eintrag $e_j^\top$, wenn $j = y + xi$. Betrachten wir die 0-ten Blocklinien, also $i = 0$ beziehungsweise $x = 0$, so verkürzt sich die Gleichung zu $j = y$ und die Blocklinien bestehen tatsächlich aus Einheitsmatrizen. Somit ist die Doppelordnung der Inzidenzmatrix gesichert. Betrachten wir das zur ersten Blockzeile gehörige lateinische Quadrat $A = (a_{y,i})$, so ist $x = 1$ und es gilt $a_{y,i} = j = y + i$. Es ist also A gerade die Cayley-Tafel der additiven Gruppe von $\mathbb{F}_n$. Ist dagegen $A_1 = (a_{x,i}^{(1)})$ das erste reduzierte lateinische Quadrat, so ist $y = 0$ und es gilt $a_{x,i}^{(1)} = j = xi$ für $x, i = 1, \ldots, n-1$. Das erste reduzierte lateinische Quadrat ist also gerade die Cayley-Tafel der multiplikativen Gruppe von $\mathbb{F}_n$. Dies erlaubt uns eine genauere Bestimmung der doppelgeordneten Inzidenzmatrizen von desarguesschen projektiven Ebenen.

Satz 7.11
Sei I eine doppelgeordnete Inzidenzmatrix einer desarguesschen projektiven Ebene der Ordnung n. Dann gilt:

i) *Die Blocklinien differieren nur in der Reihenfolge ihrer Permutationsmatrizen.*

ii) *$n = p^r$ mit einer Primzahl p.*

iii) *Die Permutationsmatrizen einer Blocklinie bilden eine zur elementarabelschen Gruppe $\mathbb{Z}_p \times \cdots \times \mathbb{Z}_p$ isomorphe Gruppe.*

iv) *Jede Permutationsmatrix des Permutationsteils hat Ordnung p und ihre Zyklen haben jeweils Länge p.*

v) *Die reduzierten lateinischen Quadrate sind alle bis auf die Nummerierung gleich und sind isomorph zur Cayley-Tafel der zyklischen Gruppe Z_{n-1}.*

Beweis:

Die meisten dieser Eigenschaften sind bereits durch Korollar 7.9 gezeigt, so daß wir nur noch nachweisen müssen, daß die durch die Permutationsmatrizen einer inneren Blocklinie gebildete Gruppe isomorph zu der elementarabelschen Gruppe ist und die reduzierten lateinischen Quadrate isomorph zur Cayley-Tafel der zyklischen Gruppe sind. Beschränken wir uns zunächst auf den Fall, daß I gemäß unserer obigen Konstruktion aus dem Galoisfeld $\mathbb{F}_n$ entstanden ist. Dann ist das durch die erste Blockzeile gegebene lateinische Quadrat gerade die Cayley-Tafel der additiven Gruppe von $\mathbb{F}_n$, welche isomorph zur elementar-abelschen Gruppe ist. Nach Lemma 7.5 bilden dann die Blöcke der ersten Blockzeile und damit auch jeder anderen inneren Blocklinie eine zu dieser elementar-abelschen Gruppe isomorphe Gruppe. Das erste reduzierte lateinische Quadrat ist dagegen isomorph zur Cayley-Tafel der multiplikativen Gruppe von $\mathbb{F}_n$, welche isomorph zur zyklischen Gruppe $\mathbb{Z}_{n-1}$ ist. Dies gilt nach Korollar 7.9 iii) dann natürlich auch für alle anderen reduzierten lateinischen Quadrate.

Kommen wir nun zum allgemeinen Fall, sei also I' eine weitere doppelgeordnete Inzidenzmatrix der desarguesschen projektiven Ebene der Ordnung n, also insbesondere $I' = PIQ^\top$ permutationsäquivalent zu I. Da die Kollineationsgruppe der desarguesschen Ebenen transitiv auf den Vierecken wirkt (vgl. Stevenson [24] Theorem 5.2.6), können wir hierbei die Permutationsäquivalenz so wählen, daß durch sie ein gegebenes Viereck festgehalten wird. Wählen wir als Viereck die Punkte p_1, p_2, p_{n+1+1} und p_{n+1+2}, die ersten beiden Spalten der Rand-Blockspalte und der 0-ten Blockspalte, so wird hierdurch die Zeile l_1, die Verbindungsgerade von p_1 und p_2, auf die Verbindungsgerade von p_1 und p_2 abgebildet, welche aufgrund der Doppelordnung gerade l_1 ist. Somit wird bei dieser Permutationsäquivalenz auch noch l_1, die erste Zeile, festgehalten und analog auch die zweite Zeile l_2, die Verbindungsgerade von p_{n+1+1} und p_{n+1+2}. Die Permutationsäquivalenz (P, Q)

fixiert also die ersten beiden Zeilen und ersten beiden Spalten. Dann besteht nach Lemma 5.2 die Wirkung der Permutationsäquivalenz auf den Kern der Inzidenzmatrix nur aus Vertauschen der inneren Blocklinien und Konjugieren der Blöcke mit einer Permutationsmatrix W.

Unter diesen Operationen werden jedoch die Permutationsmatrizen der inneren Blocklinien bis auf die Reihenfolge nur mit W konjugiert und sind daher weiterhin isomorph zur elementar-abelschen Gruppe. Für die reduzierten lateinischen Quadrate bewirkt die Multiplikation der Blöcke mit W von links ein Vertauschen der reduzierten lateinischen Quadrate untereinander, das Permutieren der Blocklinien ein Permutieren der Zeilen beziehungsweise Spalten und die Multiplikation der Blöcke mit $W^\top$ von rechts ein Umnummerieren bei den reduzierten lateinischen Quadraten. Da die reduzierten lateinischen Quadrate nach Korollar 7.9 iii) jedoch alle isomorph zueinander sind, verlassen sie durch die Permutationsäquivalenz nicht ihre Isomorphieklassen, sind also weiterhin isomorph zur Cayley-Tafel der zyklischen Gruppe $\mathbb{Z}_{n-1}$.

□

Mit diesem Satz haben wir somit unsere Vermutungen aus Kapitel 6 über die Struktur der doppelgeordneten Inzidenzmatrix zur Gänze bestätigt. Durch die vorherigen Lemmata konnten wir die Struktur von (z, A)-desarguesschen Ebenen ebenfalls genauer bestimmen, so daß sich solche Ebenen deutlich schneller berechnen lassen. Für den allgemeinen Fall, insbesondere bei kleiner Kollineationsgruppe, ergibt sich hierdurch jedoch keine Vereinfachung. Daher beschäftigen wir uns im nächsten Kapitel mit einem völlig anderen Ansatz, besonders im Hinblick auf den Nachweis der Nicht-Existenz von projektiven Ebenen bestimmter Ordnungen.

8 Blockpläne und projektive Ebenen

Dieses Kapitel handelt von einem anderen Objekt der endlichen Geometrie, nämlich den Blockplänen oder Block-Designes. Diese sind auf folgende Weise definiert (vgl. Beutelspacher [3]).

Definition 8.1
Ein (v, k, λ)**-Blockplan** ist eine Inzidenzstruktur $D = (\mathcal{O}, \mathcal{B}, I)$, wobei die Elemente von $\mathcal{O}$ **Objekte** und die Elemente von $\mathcal{B}$ **Blöcke** heißen, und es gilt:

i) D hat genau v Objekte.
ii) jeder Block von D inzidiert mit genau k Objekten, wobei $2 \leq k \leq v-1$.
iii) je zwei verschiedene Objekte lassen sich durch genau λ Blöcke verbinden.

Bei Blockplänen ist die Anzahl der Blöcke, auf denen ein festes Objekt liegt, konstant. Diese Konstante wollen wir der Literatur folgend mit r bezeichnen. Ist b die Anzahl der Blöcke, so sind r und b durch die beiden Gleichungen

$$bk = vr \tag{2}$$
$$r(k-1) = \lambda(v-1) \tag{3}$$

eindeutig durch v, k und λ bestimmt, so daß die Festlegung nur dieser Größen bei der Definition der Blockpläne Sinn macht (vgl. Hall [9]). Als Spezialfall lassen sich die projektiven Ebenen der Ordnung n als $(n^2+n+1, n+1, 1)$-Blockpläne in dieses Konzept einfügen. Wie bei den endlichen projektiven Ebenen können wir auch zu Blockplänen die Inzidenzmatrix bilden. Hierbei weisen wir jedoch den Objekten die Zeilen und den Blöcken die Spalten zu. Dann lassen sich die Eigenschaften für Blockpläne auf folgende Weise in Eigenschaften ihrer Inzidenzmatrix übertragen.

Lemma 8.2
Eine $(0,1)$-Matrix B ist genau dann eine Inzidenzmatrix eines (v, k, λ)-Blockplans D, wenn gilt:

i) *B hat genau v Zeilen.*
ii) *die Spaltensummen sind stets k, wobei $2 \leq k \leq v-2$.*
iii) *je zwei verschiedene Zeilen schneiden sich genau λ mal.*

In diesem Fall ist $r := \frac{\lambda(v-1)}{k-1}$, die Anzahl der Blöcke von D durch ein festes Objekt, gerade die Zeilensummen von B und $b := \frac{vr}{k}$, die Anzahl der Blöcke, gerade die Anzahl der Spalten von B.

Beweis:

Folgt unmittelbar aus Definition 8.1 und den Gleichungen (2) und (3).

□

Wie bereits gesehen, sind die endlichen projektiven Ebenen ein Spezialfall der Blockpläne. Wir wollen uns im folgenden jedoch eine andere Klasse von Blockplänen anschauen, welche mit projektiven Ebenen verbunden sind.

Satz 8.3
Gibt es ein System von q paarweise orthogonalen lateinischen Quadraten der Ordnung n, wobei $1 \leq q \leq n-3$, so gibt es einen $(n, q+1, q(q+1))$-Blockplan.

Beweis:

Ist $k = q+1$, so ist $2 \leq k \leq n-2$. Wir werden im folgenden eine Inzidenzmatrix eines $(n, k, k(k-1))$-Blockplans konstruieren. Hierzu formen wir die lateinischen Quadrate $L_1, \ldots, L_q$ in Blockzeilen um, wie bei der Konstruktion einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene aus einem System von $n-1$ paarweise orthogonalen lateinischen Quadraten in Kapitel 4. Fügen wir noch eine 0-te Blockzeile aus Einheitsmatrizen und einen Rand aus Stufenmatrizen hinzu, so erhalten wir eine $(0,1)$-Matrix I wie in Abbildung 14. Da nach Korollar 4.4 die Orthogonalitätsregel der lateinischen Quadrate äquivalent zur Rechtecksregel der zugehörigen Blockzeilen ist, erfüllt I die Rechtecksregel und es schneiden sich zwei verschiedene Zeilen genau einmal.

Wir bezeichnen die Blöcke mit $P_{i,j}$ wie in Abbildung 14 und ergänzen dies mit $P_{0,j} = E_n = P_{i,0}$ für $j = 0, \ldots, n-1$ und $i = 0, \ldots, q$. Dann ist die Matrix $B := \sum_{i=0}^{q} (P_{i,1}, \ldots, P_{i,n-1}) = (\sum_{i=0}^{q} P_{i,1}, \ldots, \sum_{i=0}^{q} P_{i,n-1})$ eine $(0,1)$-Matrix aufgrund der Überdeckungsfreiheit der Blöcke einer inneren Blockspalte. Man erhält B aus I, indem man die Blockzeilen, inklusive der 0-ten Blockzeile addiert und dann die Rand-Blockspalte und die 0-te Blockspalte davon abschneidet. Weiterhin ist B die gesuchte Inzidenzmatrix eines $(n, k, k(k-1))$-Blockplans, denn zunächst hat B genau n Zeilen. Die Spaltensummen sind konstant k, da wir k Blockzeilen addieren und jede Blockzeile zu jeder Spalte genau eine Eins beisteuert. Nun müssen wir noch berechnen, wie oft sich zwei verschiedene Zeilen $\hat{l}_1$ und $\hat{l}_2$ von B schneiden. Sind $L^i := \{l_0^i, \ldots, l_q^i\}$ die zugehörigen Zeilen von I aus denen $\hat{l}_i$ durch Aufsummieren und Abschneiden des Randes und der 0-ten Blockzeile entstehen für $i = 1, 2$, so schneiden sich l_i^1 mit jedem l_j^2 genau einmal. Daher schneiden sich $\sum L^1$ und $\sum L^2$ insgesamt $(q+1)^2$ mal. Allerdings schneidet sich jedes l_i^1 mit genau einem $l_j^2 \in L^2$ außerhalb der inneren Blockspalten, nämlich genau mit demjenigen l_j^2 in derselben Blockzeile wie l_i^1, was dann nichts zu den Schnitten von $\hat{l}_1$ und $\hat{l}_2$ beiträgt. Daher ist $< \hat{l}_1, \hat{l}_2 >= (q+1)^2 - (q+1) = q(q+1) = k(k-1)$. Somit ist B nach Lemma 8.2 eine Inzidenzmatrix eines $(n, k, k(k-1))$-Blockplans.

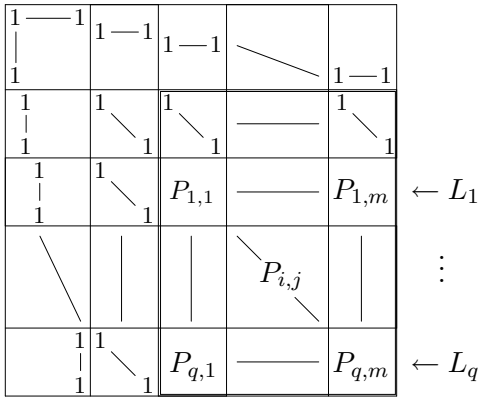


Abbildung 14: lateinische Quadrate als $(0,1)$-Matrix

□

Die Beschränkung $2 \leq q+1 = k \leq n-2$ ist lediglich durch die Definition des Blockplans in Definition 8.1 gegeben. Allgemein lassen sich mit diesem Verfahren für alle $k = 0, \ldots, n$ jeweils k Blockzeilen des Kerns einer doppelgeordneten Inzidenzmatrix einer projektiven Ebene in eine Inzidenzstruktur umwandeln, welche aus $v = n$ Objekten und $b = n(n-1)$ Blöcken besteht, wobei auf jedem Block k Objekte liegen, durch jedes Objekt $r = k(n-1)$ Blöcke gehen und sich je zwei Objekte durch genau $\lambda = k(k-1)$ Blöcke verbinden lassen. In diesem speziellen Fall sprechen wir im Sinne einer konsistenten Namensgebung auch bei $k \leq 1$ oder $k \geq n-1$ von einem $(n, k, k(k-1))$-Blockplan. Man beachte jedoch, daß in diesen Fällen r und b im Allgemeinen nicht durch v, k und λ und den Gleichungen (2) und (3) eindeutig bestimmt sind, es ist sogar nicht einmal zwingend, daß r, die Anzahl der Blöcke durch ein Objekt, für jedes Objekt gleich ist, weshalb wir dies noch extra fordern müssen. Zwei Spezialfälle ergeben sich noch, ist nämlich $k = 0$, so erhalten wir als Inzidenzmatrix gerade die Nullmatrix, ist dagegen $k = n$, so ist die Inzidenzmatrix gerade die Einsmatrix J. Aus Korollar 4.5 und Satz 8.3 ergibt sich sofort das folgende Korollar.

Korollar 8.4
Gibt es eine projektive Ebene der Ordnung n, so gibt es für alle $k = 0, \ldots, n$ einen $(n, k, k(k-1))$-Blockplan.

Interessant ist vor allem die Umkehrung dieses Korollars, denn können wir zeigen, daß es für ein $k \in \{0, \ldots, n\}$ keinen $(n, k, k(k-1))$-Blockplan gibt, so haben wir nach diesem Korollar damit gezeigt, daß es auch keine projektive Ebene der Ordnung n gibt. Hierbei ist die Inzidenzmatrix des Blockplans vom Format $n \times (n(n-1))$ und damit erheblich kleiner als die Inzidenzmatrix der projektiven Ebene, so daß eine schnellere Berechnung des Blockplans zu

erhoffen ist. Im weiteren Verlauf definieren wir zu einer $(0,1)$-Matrix I die Matrix $\bar{I}$ durch $\bar{I} := J - I$, wobei J die Einsmatrix von passenden Format sei. Dann sind bei $\bar{I}$ nur die Einsen und Nullen von I vertauscht. In manchen Fällen ist mit I auch $\bar{I}$ eine besondere Inzidenzstruktur.

Lemma 8.5
Ist B eine Inzidenzmatrix eines $(n, k, k(k-1))$-Blockplans, so ist $\bar{B}$ eine Inzidenzmatrix eines $(n, \bar{k}, \bar{k}(\bar{k}-1))$-Blockplans, wobei $\bar{k} = n - k$.

Beweis:

Sei also nun B eine Inzidenzmatrix eines $(n, k, k(k-1))$-Blockplans. Nach Lemma 8.2 sind B und damit auch $\bar{B}$ vom Format $n \times n(n-1)$. Die Spalten von $\bar{B}$ enthalten $\bar{k} = n - k$ Einsen, da B in jeder Spalte k Einsen in den n Zeilen verteilt hat. Analog enthalten die Zeilen jeweils $\bar{r} = b - r = n(n-1) - k(n-1) = (n-k)(n-1) = \bar{k}(n-1)$ Einsen.

Bleibt also noch das Skalarprodukt zweier verschiedener Zeilen von $\bar{B}$ zu berechnen. Nach Lemma 8.2 betragen die Zeilensummen von B stets $k(n-1)$. Nun ist $l + \bar{l} = e$, der Vektor aus lauter Einsen, für alle Zeilen l von B. Für je zwei verschiedene Zeilen $\bar{l}$ und $\bar{l}'$ von $\bar{B}$ ist daher $\langle \bar{l}, \bar{l}' \rangle = \langle e - l, e - l' \rangle = \langle e, e \rangle - \langle l, e \rangle - \langle e, l' \rangle + \langle l, l' \rangle = n(n-1) - k(n-1) - k(n-1) + k(k-1) = (n-k)(n-1) - k(n-k) = (n-k)(n-k-1) = \bar{k}(\bar{k}-1)$. Nach Lemma 8.2 ist B somit eine Inzidenzmatrix eines $(n, \bar{k}, \bar{k}(\bar{k}-1))$-Blockplans.

□

Nach diesem Lemma ist es sinnvoll, sich bei der Berechnung der Blockpläne auf $k \leq \frac{n}{2}$ zu beschränken. Wie bei den Inzidenzmatrizen projektiver Ebenen ist auch die Inzidenzmatrix eines Blockplans nur eindeutig bis auf Permutationsäquivalenz. Wir können daher auch die Inzidenzmatrizen der Blockpläne als doppelgeordnet annehmen, da nach A. Mader und O. Mutzbauer [18] die Existenz einer permutationsäquivalenten doppelgeordneten Form für jede $(0,1)$-Matrix gesichert ist. Hiermit läßt sich bereits der Fall $k = 2$ bestimmen.

Lemma 8.6
Es gibt für $v \geq 2$ bis auf Isomorphie nur jeweils einen $(v, 2, 2)$-Blockplan. Seine doppelgeordnete Inzidenzmatrix B_v ist eindeutig und hat folgende Form:

$$B_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$B_{v+1} = \begin{pmatrix} 1 & 1 & \cdots & \cdots & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 & & & \\ 0 & 0 & \ddots & \ddots & 0 & 0 & & B_v & \\ 0 & 0 & \cdots & 0 & 1 & 1 & & & \end{pmatrix}$$

Ist also $B_v = (b_{i,j})$, so bildet die Teilmatrix $(b_{i,j})_{2\le i\le n, 2n-1\le j\le n(n-1)}$ gerade B_{v-1} und für die restlichen Einträge gilt:

$$b_{i,j} = \begin{cases} 1 & \text{für } i = 1 \text{ und } j \le 2(n-1) \text{ oder} \\ & \text{für } i \ge 2 \text{ und } j \in \{2(i-1)-1, 2(i-1)\} \\ 0 & \text{sonst.} \end{cases}$$

Beweis:

Wir verwenden die Bezeichnungen von Definition 8.1. Dann gilt für den $(v,2,2)$-Blockplan $k = 2, \lambda = 2, b = v(v-1)$ und $r = 2(v-1)$. Sei B_v eine doppelgeordnete Inzidenzmatrix eines Blockplans mit diesen Parametern. Ist $v = 2$ erhalten wir für den $(2,2,2)$-Blockplan die Inzidenzmatrix $B_2 = J = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Bei $B_v, v > 2$ haben wir in einer $(v \times v(v-1))$-Matrix Einsen und Nullen so zu verteilen, daß die Zeilensummen jeweils $r = 2(v-1)$ und die Spaltensummen jeweils $k = 2$ ergeben, wobei je zwei verschiedene Zeilen sich genau zweimal schneiden. Durch die Doppelordnung bekommt B_v dann die Form

$$B_v = \begin{pmatrix} 1 & 1 & \cdots & \cdots & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 & & & \\ 0 & 0 & \ddots & \ddots & 0 & 0 & & A & \\ 0 & 0 & \cdots & 0 & 1 & 1 & & & \end{pmatrix}.$$

Man beachte hierbei, daß die Form der ersten $r = 2(v-1)$ Spalten bereits durch die Ordnung der Spalten bestimmt ist. Nun ist A eine $(0,1)$-Matrix vom Format $((v-1) \times (v-1)(v-2))$, wobei die Zeilensummen jeweils $2(v-1) - 2 = 2(v-2)$ und die Spaltensummen 2 betragen. Zudem, da sich die Zeilen von A nicht außerhalb von A schneiden, müssen sie sich jeweils zweimal innerhalb von A schneiden. A erfüllt somit die Bedingungen von B_{v-1} und ist daher permutationsäquivalent dazu. Nun müssen jedoch durch die Doppelordnung von B_v zumindest die Spalten von A geordnet sein. Hierdurch ist jedoch A bereits doppelgeordnet, wie wir induktiv gesehen haben. Daher ist $A = B_{v-1}$ und die doppelgeordneten Inzidenzmatrizen von $(v,2,2)$-Blockplänen haben obige Form.

□

Dieses Lemma zeigt insbesondere, daß $(v,2,2)$-Blockpläne für alle $v \ge 2$ existieren. Dies ist nach Satz 8.3 wenig überraschend, gibt es doch zu jeder Ordnung zumindest ein lateinisches Quadrat. Wir wollen uns nun jedoch der Anwendung der Doppelordnung auf die computerunterstützte Berechnung von Blockplänen zuwenden.

Unser Programm-Paket blockplan berechnet alle doppelgeordneten Inzidenzmatrizen von (v,k,λ)-Blockplänen mithilfe eines modifizierten Backtracking-Verfahrens. Hierzu geben wir anfangs die ersten beiden Zeilen vor,

welche bereits durch die Doppelordnung eindeutig festgelegt sind, und versuchen nun jeweils eine Zeile anzufügen, welche bezüglich der lexikographischen Ordnung nicht größer als die vorherige Zeile ist und bezüglich der vorigen Zeilen auch keinen Widerspruch mit den Regeln des Blockplans aufweist. Erreichen wir hiermit die letzte Zeile, so erhalten wir eine doppelgeordnete Inzidenzmatrix eines Blockplans. Das Programm ist in folgende Dateien aufgeteilt:

Main.cpp
Das Hauptprogramm. Es enthält alle für die Berechnung wichtigen Funktionen.

getclock.h
Funktionen von F. Schmitt zur Auswertung der benötigten Prozessorzeit.

newmath.h
Funktionen von M. Baumann zur Speicherung einer Matrix im pbm-Format.

Bevor wir uns den einzelnen Funktionen zuwenden, besprechen wir noch die wichtigsten Variablen und auf welche Weise sie die gesuchte Inzidenzmatrix speichern.

inzMat: Dieses zweidimensionale Array enthält die gesuchte Inzidenzmatrix, indem zu jeder Zeile und zu jeder Eins dieser Zeile die Spaltennummer gespeichert ist.

blockMat: Enthält dual zu inzMat zu jeder Spalte die Zeilen, welche bereits mit Einsen belegt sind.

weigBlock: Kontrolliert zu jeder Spalte, wie viele Einsen bereits darin gesetzt wurden.

schnitt: Enthält jederzeit zu jeder Zeile, wie oft sie die Zeilen mit kleinerem Index aufgrund der bereits gesetzten Einsen schneidet.

zeile: Die aktuell zu besetzende Zeile.

Weiterhin sind in den Konstanten V, K und LAMBDA die Parameter v, k und λ des Blockplans gespeichert und können den Bedürfnissen gemäß festgelegt werden. Die anderen Konstanten R und B, die Anzahl der Einsen auf einer Zeile und die Anzahl der Spalten werden hieraus selbständig berechnet, falls $k \geq 2$. Doch kommen wir nun zu den einzelnen Funktionen des Programms.

main()
Diese Funktion steuert die Berechnung des Blockplans. Hierzu werden zunächst die ersten beiden Zeilen der Inzidenzmatrix vorgegeben. Falls möglich wird noch die letzte zwischengespeicherte Situation in einlese() eingelesen und von dieser aus weitergerechnet. Die eigentlichen Rechnungen werden in nextOb() durchgeführt, wo jeweils die nächste mit den Regeln bezüglich

den bisherigen Zeilen widerspruchsfrei setzbare Zeile ermittelt wird. Um das Einlesen der letzten berechneten Situation zu ermöglichen, wird zudem noch in zwischenSpeichern() diese jeweils gespeichert.

nextOb()

In dieser Funktion wird die lexikographisch nächstkleinere Zeile berechnet, die sich im Einklang mit den Regeln bezüglich der vorherigen Zeilen setzen läßt. Durch den booleschen Parameter vor läßt sich die Funktion auf zwei verschiedene Weisen aufrufen. Ist nämlich vor wahr, so wollen wir an die bisherigen Zeilen eine weitere Zeile anfügen. Daher setzen wir als Ausgangszeile gerade die Zeile darüber, da aufgrund der Doppelordnung die Zeilen mit zunehmenden Index nicht lexikographisch größer werden. Ist dagegen vor falsch, so hat diese Zeile zu keinem Ergebnis geführt und wir wollen daher diese Zeile durch die nächstkleinere mit den Regeln konforme Zeile ersetzen.

Zur Berechnung dieser nächstkleineren Zeile erhöhen wir von hinten kommend jeweils den letzten Eintrag der durch zeile gegebenen Zeile von intMat, also die Spaltennummer der letzten Eins der aktuellen Zeile der Inzidenzmatrix, und überprüfen dann jeweils mithilfe der Funktion eintragGueltig(), ob diese gesetzte Eins in Einklang mit den Regeln des Blockplans steht. Finden wir so einen widerspruchsfreien Eintrag, so wiederholen wir dies mit der nächsten zu setzenden Eins in dieser Zeile. Läßt sich dagegen die Eins an keiner Stelle setzen, ohne den Regeln des Blockplans zu widersprechen, so löschen wir die letzte Eins und wiederholen dies mit der jetzt neuen letzten Eins der Zeile. Gelingt es uns hiermit, insgesamt R Einsen in der Zeile widerspruchsfrei zu setzen, so versuchen wir durch erneutes Aufrufen dieser Funktion, eine weitere Zeile an die bisherige $(0,1)$-Matrix anzufügen. Führen dagegen alle Einträge zu Widersprüchen, so löschen wir diese Zeile und berechnen durch erneutes Aufrufen dieser Funktion zur vorherigen Zeile die nächstkleinere konforme.

eintragGueltig()

In dieser Funktion ermitteln wir zunächst, ob die Spaltensumme durch die neu gesetzte Eins nicht größer als K und ob der Schnitt mit den vorherigen Spalten nicht größer als LAMBDA wird. Kommt es hier zu keinem Widerspruch, überprüfen wir, ob der entsprechende Spaltenvektor, der die neue Eins enthält, lexikographisch nicht größer ist als der Spaltenvektor links davon, um der Doppelordnung nicht zu widersprechen. Einen weiteren Test führen wir in dem Fall durch, wenn der Spaltenindex der neuen Eins größer ist als die Indizes aller Einsen einer vorherigen Zeile. In diesem Fall tragen alle weiteren noch in der aktuellen Zeile zu setzenden Einsen nichts mehr zum Schnitt mit dieser vorigen Zeile bei, weshalb der Schnitt bereits jetzt schon LAMBDA betragen muß.

zwischenSpeichern()

Zur Zwischenspeicherung sichern wir abwechselnd in den Dateien ,,speicher.z1“ und ,,speicher.z2“ die bisher berechnete Konfiguration. Hierzu schreiben wir die Inzidenzmatrix, wie sie durch inzMat gegeben ist, zusammen mit dem Index der aktuell betrachteten Zeile und der Anzahl der bisher bereits berechneten Inzidenzmatrizen von Blockplänen in die Datei. In der Funktion einlese() erkennt dann das Programm selbständig, welche dieser Dateien die am weitesten fortgeschrittene Konfiguration enthält, und setzt den Algorithmus mit dieser Konfiguration fort.

Das Programm läßt sich wiederum durch den Befehl ,,g++ Main.cpp -O“ mit dem GNU-C++-Compiler unter Linux compilieren und erzeugt dann die Datei ,,a.out“. Aufgerufen wird das Programm dann durch ,,a.out“ und erzeugt neben den Dateien ,,zwischen.z1“ und ,,zwischen.z2“ zum Zwischenspeichern noch die Datei ,,ausgabe.txt“ für die Ausgabe der benötigten Rechenzeit und natürlich die Dateien ,,blockv_k_λ__x.pbm“, welche die berechneten Inzidenzmatrizen der Blockpläne im pbm-Format enthalten. Hierbei sind v, k und λ durch die Parameter des (v, k, λ)-Blockplans vorgegeben und x ist die Nummer des Blockplans.

Neben dem schönen Resultat, nun ein Werkzeug zur Berechnung beliebiger Blockpläne zur Hand zu haben, sind natürlich die Ergebnisse in Hinblick auf die Existenz von projektiven Ebenen bestimmter Ordnung von besonderer Bedeutung. Der erste interessante Fall diesbezüglich ergibt sich für $v = 6$ und $k = 3$, gibt es doch nach Tarry [25] keine zwei orthogonalen lateinischen Quadrate der Ordnung 6, so daß die Frage der Existenz eines $(6, 3, 6)$-Blockplans nicht bereits durch Satz 8.3 beantwortet ist. Ein Test durch unser Programm zeigt jedoch die Existenz eines $(6, 3, 6)$-Blockplans. Die Umkehrung von Satz 8.3 gilt somit nicht, es kommt also durchaus vor, daß es einen $(v, k, k(k-1))$-Blockplan gibt, jedoch keine $k-1$ paarweise orthogonalen lateinischen Quadrate der Ordnung v. Dies ist ein zunächst ernüchterndes Ergebnis, zeigt sich doch unsere Reduktion auf den Blockplan als weniger effektiv in Bezug auf den Ausschluß der Existenz von projektiven Ebenen bestimmter Ordnungen als erhofft. Der nächste interessante Fall ist $v = 10$, ist doch für niedrigeres v die Existenz des Blockplans durch Satz 8.3 geklärt. Hierbei berechnete unser Programm $(10, k, k(k-1))$-Blockpläne für $k = 3$ und $k = 4$, allerdings ist die Berechnung bereits so aufwendig, daß eine vollständige Behandlung dieser Blockpläne nicht mit diesem Programm durchführbar scheint. Einen Testlauf führten wir auch für $k = 5$ durch, bislang jedoch ohne Ergebnis, so daß wir diesen Fall erst nach Bereitstellung effektiverer Methoden zur Berechnung von Blockplänen klären können.

Zusammenfassend zeigt sich hiernach unser Konzept als noch nicht ausreichend um einen zu Lam [16] alternativen computerunterstützten Beweis für die Nicht-Existenz einer projektiven Ebene der Ordnung 10 anzubieten. Die Probleme scheinen hierbei einmal von geschwindigkeitstechnischer Natur

zu sein, aber auch von fundamentaler, existiert doch ein $(6, 3, 6)$-Blockplan, obwohl es keine zwei orthogonalen lateinischen Quadrate der Ordnung 6 gibt. Eine Verkürzung der Rechenzeit läßt sich sicherlich durch eine geeignetere Datenstruktur, aber auch durch eine effektivere Beschränkung der jeweils berechneten Teilmatrizen auf eine je Permutationsäquivalenzklasse erreichen. Die Tatsache, daß es anscheinend häufig vorkommt, daß ein $(v, k, k(k-1))$-Blockplan existiert, obwohl es keine $k-1$ paarweise orthogonalen lateinischen Quadrate der Ordnung v gibt, läßt dagegen darauf schließen, daß wir durch den Übergang auf die Blockpläne zu viele der Einschränkungen, wie sie bei der projektiven Ebene gegeben sind, verlieren. Es ist daher naheliegend sich weitere Bedingungen an die Blockpläne zu überlegen, die sie erfüllen müssen, wenn sie von einer projektiven Ebene gemäß unserer Transformation herrühren.

Eine dieser Eigenschaften zeigt sich, wenn wir uns vorstellen, daß wir die ersten k Blockzeilen in einen Blockplan mit Inzidenzmatrix B umwandeln und dann die nächsten k' Blockzeilen in einen weiteren Blockplan mit Inzidenzmatrix B'. In diesem Fall stehen die Matrizen B und B' in besonderer Beziehung zueinander. So sind B und B' überdeckungsfrei und eine Zeile von B schneidet sich mit einer Zeile von B', welche anderen Zeilenindex hat, genau kk' mal. Als Folge hiervon ist die Summe der Matrizen $B + B'$ eine Inzidenzmatrix eines $(v, \tilde{k}, \tilde{k}(\tilde{k}-1))$-Blockplans, wobei $\tilde{k} = k + k'$ ist. Von dieser letzten Aussage gilt auch die Umkehrung, d. h. ist $\tilde{B}$ die Inzidenzmatrix eines $(v, \tilde{k}, \tilde{k}(\tilde{k}-1))$-Blockplans, welcher gemäß unserer Konzeption von einer Inzidenzmatrix einer projektiven Ebene herrührt, so ist $\tilde{B} = B + B'$ mit B, der Inzidenzmatrix eines $(v, k, k(k-1))$-Blockplans, und B', der Inzidenzmatrix eines $(v, k', k'(k'-1))$-Blockplans, wobei $\tilde{k} = k + k'$ ist und die Matrizen B und B' ebenfalls von einer projektiven Ebene herrühren.

Dies läßt sich beispielsweise nutzen, indem man einem Blockplan B jeweils noch einen weiteren Blockplan B' anfügt, so daß die beiden Blockpläne die erwähnten Beziehungen zueinander haben. Ein erstes Testprogramm, welches an einen $(10, 2, 2)$-Blockplan noch einen $(10, 1, 0)$-Blockplan anzuhängen versuchte, ergab jedoch bereits so zahlreiche Inzidenzmatrizen, daß eine Behandlung aller dieser nicht erfolgversprechend ist, ohne eine intensive Reduktion der Ergebnisse, beispielsweise durch Nutzung von zusätzlichen Permutationsäquivalenzen, anzuwenden. Eine andere Idee ist, einen $(v, \tilde{k}, \tilde{k}(\tilde{k}-1))$-Blockplan mit Inzidenzmatrix $\tilde{B}$ in einen $(v, k, k(k-1))$-Blockplan mit Inzidenzmatrix B und einen $(v, k', k'(k'-1))$-Blockplan mit Inzidenzmatrix B' aufzuspalten, so daß $\tilde{B} = B + B'$ gilt. Dies läßt sich fortführen, bis die Matrix $\tilde{B}$ in lauter Inzidenzmatrizen von $(v, 1, 0)$-Blockplänen aufgespaltet ist. Hierbei gilt es noch, geeignete Kriterien zu erarbeiten, welche eine solche Aufspaltung verhindern, so daß hierdurch im Idealfall bereits bei der Berechnung der Inzidenzmatrizen viele davon aussortiert werden können. Gelingt eine effektive Aufspaltung der Blockpläne in lauter $(v, 1, 0)$-Blockpläne, so ist es zudem kein großer Schritt mehr zurück

zur Inzidenzmatrix der projektiven Ebene und damit zu dem großen Fernziel, einen noch effektiveren Algorithmus zur Berechnung von endlichen projektiven Ebenen zu erstellen.

9 Ausblick

In dieser Arbeit konnten wir mit Hilfe der Doppelordnung eine effektive Methode anbieten, bis Ordnung 9 alle projektiven Ebenen zu berechnen. Zudem konnten wir zusätzlich die Doppelordnung zu einem vergleichsweise schnellen Test der Inzidenzmatrizen auf Permutationsäquivalenz nutzen, welcher sich sicherlich auch noch für Ebenen höherer Ordnung verwenden ließe. Die genutzten Methoden reichen jedoch noch nicht, um die Existenzfrage für höhere Ordnungen, insbesondere Ordnung 10 klären zu können. Hierfür gibt es jedoch noch zahlreiche andere erfolgversprechende Ansätze.

Der am schnellsten zu verwirklichende Ansatz ist sicherlich, sich auf (z, A)-desarguessche Ebenen zu beschränken. Für diese Ebenen haben wir in Kapitel 7 bereits starke Beschränkungen bezüglich ihrer doppelgeordneten Inzidenzmatrix zusammengetragen, so daß sich ihre Berechnung, wie bereits erwähnt, wesentlich schneller gestalten läßt. Besonders für (z, A)-desarguessche Ebenen, bei denen das Zentrum auf der Achse liegt, ist damit eine starke Beschleunigung zu erwarten.

Für den allgemeinen Fall bieten die Blockpläne aus Kapitel 8 einen erfolgversprechenden Ansatz, insbesondere, wenn es darum geht, die Existenz einer projektiven Ebene auszuschließen. Hier ist natürlich der Algorithmus noch zu beschleunigen, vielleicht gar eine geeignetere Vorstrukturierung der Inzidenzmatrizen anzuwenden. Dazu bleibt noch zu untersuchen, welche weiteren Kriterien sich für die Blockpläne ergeben, welche von projektiven Ebenen herrühren, so daß sich hieraus effektivere Methoden zur Berechnung speziell dieser Blockpläne erarbeiten lassen. Ob sich dieses Konzept gar zu einen neuen, effektiveren Algorithmus zur Berechnung der zugehörigen Inzidenzmatrix einer projektiven Ebene ausbauen läßt, wird sich allerdings erst in Zukunft klären lassen.

Ein weiterer Ansatz ist es, von der doppelgeordneten Inzidenzmatrix den Rand wegzulassen und nur den Kern zu betrachten, was wir mit dem Begriff Schälen bezeichnen wollen. Dieses Schälen können wir mit dem Kern wiederholen, indem wir nun ihn doppelordnen und den Rand aus Stufenmatrizen entfernen und erhalten so einen Kern 2. Ordnung. Führen wir nun dieses Schälen sukzessive fort, so erhalten wir Kerne immer höherer Ordnung, welche als Teilmatrizen der gesamten Inzidenzmatrix ebenfalls die Rechtecksregel erfüllen. Da sie zudem immer kleiner werden, ist eine schnellere Berechnung zu erhoffen. Hier sind noch genauere Eigenschaften dieser Kerne zu untersuchen, insbesondere gilt es, notwendige Kriterien zu entwickeln, welche die umgekehrte Richtung zu den Kernen immer kleinerer Ordnung bis hin zur Inzidenzmatrix der projektiven Ebene ermöglichen.

Somit sind, wie bei den meisten wissenschaftlichen Arbeiten, noch viele Fragen offen und neue Fragen hinzugekommen. Dieses Thema bietet daher genügend Raum auch für künftige Forschungsprojekte.

Literatur

[1] R. Baer, *Homogenity of projective planes*, Amer. Journ. of Math., Vol. 64 (1942), S. 137–152.

[2] D. Betten, *Zum Satz von Euler-Tarry*, MNU, 36, Heft 8 (1983), S. 449–453.

[3] A. Beutelspacher, *Einführung in die endliche Geometrie I / II*, Bibliographisches Institut, Mannheim Wien Zürich, 1982 / 1983.

[4] R. C. Bose, *On the application of the properties of Galois fields to the problem of construction of hyper-Graeco-Latin squares*, Sankhyã, Vol. 3 (1938), S. 323–338.

[5] I. N. Bronstein, K. A. Semendjajew, G. Musiol, H. Mühlig, *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun und Frankfurt am Main, 1995 (2).

[6] R. H. Bruck, H. J. Ryser, *The nonexistence of certain projective planes*, Can. Journ. of Math., Vol. 1 (1949), S. 88–93.

[7] R. C. Brualdi, H. J. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, 1991.

[8] J. H. Conway, R. K. Guy, *The book of numbers*, Springer-Verlag, New York, 1996.

[9] M. Hall, Jr., *Combinatorial Theory, Second Edition*, Wiley-Interscience Publication, 1986.

[10] J. Huber, *Inzidenzmatrizen endlicher projektiver Ebenen*, Zulassungsarbeit (Würzburg), 2000.

[11] N. Jacobson, *Basic Algebra I / II*, Freeman, San Francisco, 1974 / 1980.

[12] F. Kárteszi, *Introduction to finite geometries*, North-Holland Publishing Company Amsterdam Oxford, 1976.

[13] F. Knoflíček, *A combinatorial approach to the known projective planes of order nine*, Mathematica Bohemica, Vol. 120 (1995), No. 4, S. 347–366.

[14] R. Kochendörffer, *Lehrbuch der Gruppentheorie unter besonderer Berücksichtigung der endlichen Gruppen*, Akademische Verlagsgesellschaft Geest & Partig K. G., Leipzig, 1966.

[15] C. W. H. Lam, G. Kolesova, L. Thiel, *A computer search for finite projective planes of order* 9, Discrete Mathematics, Vol. 92 (1991), S.187–195.

[16] C. W. H. Lam, *The Search for a finite projective plane of order* 10, Amer. Math. Monthly, Vol. 98 (1991), S. 305–318.

[17] R. Lingenberg, *Grundlagen der Geometrie*, Bibliographisches Institut, Mannheim Wien Zürich, 1978 (3).

[18] A. Mader, O. Mutzbauer, *Doubleordering of* $(0,1)$*-matrices*, Ars Combinatorica (2000), S. 81–95.

[19] M. Mann, *Berechnung von Inzidenzmatrizen endlicher projektiver Ebenen*, Diplomarbeit (Würzburg), 2001.

[20] G. Pickert, *Projektive Ebenen*, Springer-Verlag, Berlin Heidelberg New York, 1975 (2).

[21] H. Schildt, *C++ Ent-Packt*, mitp-Verlag, Landsberg, 2001.

[22] U. Schöning, *The graph isomorphism problem*, Birkhäuser, Boston, 1993.

[23] B. Schwarz, *Permutationsäquivalenzklassen von* $(0,1)$*-Matrizen*, Diplomarbeit (Würzburg), 2001.

[24] F. W. Stevenson, *Projective Planes*, Freeman, San Francisco, 1972.

[25] M. G. Tarry, *Le problème des* 36 *officiers*, C. R. Assoc. Fran. Av. Sci. Vol. 1 (1900), S. 122–123, Vol. 2 (1901), S. 170–203.

[26] K. W. Wagner, *Theoretische Informatik*, Springer-Verlag, Berlin Heidelberg, 2003 (2).

Index