Computergestützte Suche nach optimalen linearen Codes über endlichen Kettenringen unter Verwendung heuristischer Methoden

Der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
vorgelegte Abhandlung

von
Johannes Zwanzger
geboren am 15.03.1982 in Forchheim

1. Gutachter:	Prof. Dr. Adalbert Kerber
2. Gutachter:	Prof. Dr. Patric Östergård
3. Gutachter:	Prof. Dr. Michael Stoll

Tag der Einreichung:	21. 01. 2011
Tag des Kolloquiums:	27. 05. 2011

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Liste der Algorithmen

Symbolverzeichnis

$\mathbb{N}$	Die Menge der natürlichen Zahlen mit Null, 1
$\mathbb{N}^*$	Die Menge der natürlichen Zahlen ohne Null, 1
$\mathbb{P}$	Die Menge aller Primzahlen, 2
R	Ein Ring mit Eins, normalerweise ein endlicher Kettenring, 1
(T)	Das von der Menge $T \subset R$ erzeugte beidseitige Ideal, 1
$\mathrm{Rad}(R)$	Das Jacobson-Radikal von R, 1
$R^\times$	Die Einheitengruppe von R, 2
m	Die Kettenlänge von R, 2
p,r	$p^r = q$, 2
q	Primzahlpotenz, so dass $\mathbb{F}_q \cong R/\mathrm{Rad}(R)$, 2
$\mathbb{F}_q$	Der endliche Körper mit q Elementen, für eine Primzahlpotenz q, 2
θ	Ein beliebig, aber fest gewählter Erzeuger von $\mathrm{Rad}(R)$, 2
$\mathfrak{h}(x)$	Die Höhe des Elementes $x \in R$, 3
$\mathcal{A}$	Eine Teichmüllermenge von R, 3
$\mathcal{A}^*$	$\mathcal{A} \setminus \{0\}$, 3
α	Ein Erzeuger von $\mathcal{A}^*$, 3
$f^C_{p,r}$	Das klassische Conwaypolynom vom Grad r über $\mathbb{F}_p$, 7
$f^C_{p,r,m}$	Verallgemeinertes Conwaypolynom vom Grad r über $\mathbb{Z}_{p^m}$, 8
$\mathcal{B}$	Basis eines R-linearen Codes, 14
I_k	Die Einheitsmatrix der Dimension $k \times k$, 14
λ	$\lambda = (\lambda_0, \ldots, \lambda_{k-1})$, der Umriss eines Codes, 16
$\mathfrak{h}(u)$	Die Höhe eines Vektors $u \in R^k$, 16
$\mathcal{V}_\lambda$	Spaltenmenge zum Umriss λ, 16
$\mathcal{U}_\lambda$	Menge der Informationsvektoren zum Umriss λ, 16
${\mathcal{V}_\lambda}^*$	$\mathcal{V}_\lambda \setminus \{0\}$, 24
${\mathcal{U}_\lambda}^*$	$\mathcal{U}_\lambda \setminus \{0\}$, 24
$\overline{\mathcal{V}_\lambda}$	Repräsentantenmenge von ${\mathcal{V}_\lambda}^*$ unter Rechtsmult. mit Einheiten, 24
$\overline{\mathcal{U}_\lambda}$	Repräsentantenmenge von ${\mathcal{U}_\lambda}^*$ unter Linksmult. mit Einheiten, 24
$\mathfrak{v}$	$\lvert\overline{\mathcal{V}_\lambda}\rvert$, 24
$\mathfrak{u}$	$\lvert\overline{\mathcal{U}_\lambda}\rvert$, 24
Γ_x	Die zum Vektor $x \in \mathbb{N}^{\mathfrak{v}}$ gehörende Generatormatrix, 24
x_Γ	Der zur Generatormatrix Γ gehörende Vektor $x \in \mathbb{N}^{\mathfrak{v}}$, 24
$\mathcal{M}^w$	Vgl. System (2.3), 25
$\iota(v)$	Die erste Position mit Höhe $\mathfrak{h}(v)$ im Vektor $v \in R^k$, 25
φ_λ	R-Linksmodulisomorphismus von $\mathcal{V}_\lambda$ nach $\mathcal{U}_\lambda$, 26
$\overline{\mathcal{V}_\lambda}(\vartheta, \zeta)$	$\{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(v) = \vartheta, \iota(v) = \zeta\}$, 27
$\mathrm{PG}(k-1, q)$	Die projektive Geometrie der Dimension $k-1$ über $\mathbb{F}_q$, 27

$\mathbf{p}(v)$	$\langle v\rangle_R$, der zum Vektor v gehörende Punkt, 27
$\mathbf{h}(\overline{u}) = u^\perp$	$\{v \in R^k : uv = 0\}$, die zum Vektor u gehörende Hyperebene, 27
$\mathrm{PHG}(k-1, R)$	Die projektive Hjelmslev-Geometrie der Dimension $k-1$ über R, 28
$\mathcal{G}_n$	$R^\times \wr_n S_n$, die Gruppe monomialer Transformationen, 30
$\mathcal{H}_n$	$(R^\times)^n \rtimes (S_n \times \mathrm{Aut}(R))$, die semimonomiale Transformationsgruppe, 32
G_n^λ	Menge der Generatormatrizen mit n Spalten zum Umriss λ, 34
$\mathcal{A}^\lambda$	Vgl. Definition 2.25, 35
$\mathcal{S}^\lambda$	Vgl. Lemma 2.23, 36
$\tau_i^{\overline{u}}$	$\lvert\{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(\overline{u}v) = i\}\rvert$ für $\overline{u} \in \overline{\mathcal{U}_\lambda}$, die Höhenverteilung der Zeile $\overline{u}$, 40
$\omega_i^{\overline{u}}$	$\lvert\{v \in \overline{\mathcal{V}_\lambda} : w(\overline{u}v) = i\}\rvert$ für $\overline{u} \in \overline{\mathcal{U}_\lambda}$, Gewichtsverteilung der Zeile $\overline{u}$, 40
$\sigma_{\overline{u}}(x)$	Gewichtssumme von x in Zeile $\overline{u}$, 42
$\mathcal{N}(x)$	Menge der zulässigen Nachfolger von x nach Kapitel 3, 42
$\mathcal{S}_{\overline{u}}(x)$	Lösungsmenge zu x für Zeile $\overline{u}$, 43
$\delta_{\overline{u}}(x)$	Lösungsdichte zu x für Zeile $\overline{u}$, 43
$\mathcal{E}(x)$	$\prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \delta_{\overline{u}}(x)$, die Bewertungsfunktion aus Kapitel 3, 43
$\epsilon_{\overline{u}}(n', d')$	Vorberechnete Tabellenwerte. $\epsilon_{\overline{u}}(n - \mathbb{1}^T x, d - \sigma_{\overline{u}}(x)) = \delta_{\overline{u}}(x)$, 43
$\mathcal{E}(\Gamma)$	$\mathcal{E}(x_\Gamma)$, 51
$\mathcal{L}_A^\sigma$	$\mathcal{V}_\lambda \to \mathcal{V}_\lambda, v \mapsto \underline{A\sigma(v)}$, 52
$\mathcal{R}_A^\sigma$	$\mathcal{U}_\lambda \to \mathcal{U}_\lambda, \overline{u} \mapsto \overline{\sigma(u)A}$, 52
$\mathrm{Aut}(\mathcal{S})$	$\{(A, \sigma) \in \mathrm{GL}(k, q) \rtimes \mathrm{Aut}(R) : (A, \sigma) * \mathcal{S} = \mathcal{S}\}$, 59
$\mathcal{M}_G^w$	Die Kramer-Mesner-Matrix zur Gruppe G und zum Gewicht w, 60
$\varrho_V^l(v)$	Repräsentant aus $\overline{\mathcal{V}_\lambda}$ bzgl. Linksmult. mit Einheiten zu $v \in {\mathcal{V}_\lambda}^*$, 61
$\varrho_V^r(v)$	Repräsentant aus $\overline{\mathcal{V}_\lambda}$ bzgl. Rechtsmult. mit Einheiten zu $v \in {\mathcal{V}_\lambda}^*$, 61
$\varrho_U^l(\overline{u})$	$\varphi_\lambda \circ \varrho_V^l \circ \varphi_\lambda^{-1}(\overline{u})$, Repräsentant aus $\overline{\mathcal{U}_\lambda}$ bzgl. Linksmult. mit Einh., 61
χ	$R \to \{X_0, X_1, \ldots, X_m\}$, $r \mapsto X_{m-\mathfrak{h}(r)}$, das symmetrisierte Gewicht, 67
$\mathcal{T}$	Vgl. System (4.3), 70
$\mathscr{N}(x)$	Menge der zulässigen Nachfolger von x nach Kapitel 4, 71
$\mathscr{S}_i(x)$	Lösungsmenge zu x für Zeile i, 71
$\delta_i(x)$	Lösungsdichte zu x für Zeile i, 71
$\mathscr{E}(x)$	$\prod_{i=0}^{k-1} \delta_i(x)$, die Bewertungsfunktion aus Kapitel 4, 71
$\epsilon_i(n', s)$	Vorberechnete Tabellenwerte. $\epsilon_i(n - \omega^T x, \mathcal{T}_{i*}x) = \delta_i(x)$, 72
$\mathrm{Aut}_{\mathrm{inn}}(R)$	Die Gruppe der inneren Automorphismen von R, 77
$S_{\mathcal{A}^*}$	$\{\sigma \in \mathrm{Aut}(R) : \sigma(\mathcal{A}^*) = \mathcal{A}^*\}$, 77
S_α	$\{\sigma \in \mathrm{Aut}(R) : \sigma(\alpha) = \alpha\}$, 77
$\mathfrak{p}$	$\{\overline{s} \in \mathbb{Z}_r : \exists \sigma \in S_{\mathcal{A}^*} : \sigma(\alpha) = \alpha^{(p^s)}\}$, 78
$u_\mathfrak{p}$	$\min\{s \in \{1, 2, \ldots, r\} : \overline{s} \in \mathfrak{p}\}$, 78

Vorwort

In den Jahren 1968 und 1972 entdeckten Preparata [Pre68] bzw. Kerdock [Ker72] zwei unendliche Serien sehr guter nichtlinearer binärer Codes. Beide umfassen den *Nordstrom-Robinson-Code* [NR67], einen $(16, 2^8, 6)$-Code[1], dessen Minimaldistanz die obere Schranke von 5 für *lineare* binäre Codes gleicher Länge und Kardinalität übertrifft. Lange Zeit war unklar, warum die Codes beider Serien formal dual zueinander sind, d. h. warum ihre Gewichtszähler die MacWilliams-Identität erfüllen. Erst in den neunziger Jahren fand man heraus, dass sie als Bilder linearer Codes über dem Ring $\mathbb{Z}_4$ unter der sogenannten *Grayabbildung* dargestellt werden konnten [Nec91, HKC$^+$94]. Diese Entdeckung löste einerseits das Rätsel und rückte gleichzeitig die Untersuchung linearer Codes über $\mathbb{Z}_4$ in den Fokus der Forschung. In den Folgejahren wurden Codes über endlichen Kettenringen als natürliche Verallgemeinerung der klassischen Codes über endlichen Körpern erkannt.

Für jeden endlichen Kettenring R existiert ein Restklassenkörper $\mathbb{F}_q \cong R/\mathrm{Rad}(R)$, und mit Hilfe einer verallgemeinerten Version der Grayabbildung [GS99] kann jeder R-lineare Code in einen - für gewöhnlich nichtlinearen - Code über $\mathbb{F}_q$ überführt werden. R-lineare Codes, deren Graybild eine bessere Minimaldistanz aufweist als optimale lineare Codes über $\mathbb{F}_q$ mit denselben Parametern, nennen wir *BTL*-Codes („better-than-linear“). Ist noch unklar, ob lineare Codes derselben Minimaldistanz über $\mathbb{F}_q$ existieren, sprechen wir von *BTKL*-Codes („better-than-known-linear“). Im Unterschied zu den umfassenden Tabellen für lineare Codes über Körpern gab es - abgesehen von $\mathbb{Z}_4$ [Asa] - bisher nur wenig vergleichbares Datenmaterial zu linearen Codes über endlichen Kettenringen. Diese Lücke zu schließen und gleichzeitig nach weiteren Beispielen für BTL- und BTKL-Codes zu suchen, waren die Hauptziele der vorliegenden Arbeit.

Um dies zu erreichen, wurde ein heuristischer Algorithmus aus meiner Diplomarbeit [Zwa07] für die Suche nach guten linearen Codes über endlichen Körpern auf die Situation über endlichen Kettenringen verallgemeinert. Es handelt sich hierbei um einen Greedy-Algorithmus, der versucht, die gewünschten Codes durch schrittweises Erweitern von Generatormatrizen zu konstruieren. Die Entscheidungen in jedem Schritt basieren dabei auf einer von probabilistischen Überlegungen geleiteten Bewertungsfunktion. Eine weitere Verallgemeinerung ermöglichte es außerdem, die Methode auf eine größere Klasse von Problemen anzuwenden. In dieser Arbeit betraf dies im Speziellen die Konstruktion linearer Codes nach der Kramer-Mesner-Methode (vgl. [KM76], [Bra04], [BKW05]), also solchen, deren Automorphismengruppe eine bestimmte, vorgeschriebene Untergruppe enthält.

[1] Einen Blockcode der Länge n, Kardinalität s und Minimaldistanz d über einem endlichen Körper bzw. Ring R bezeichnen wir hier wie im Folgenden kurz als (n, s, d)-Code über R. Die zugrundeliegende Metrik wird stets aus dem Zusammenhang klar sein.

Mit Hilfe dieser Verfahren wurde eine Datenbank [Zwa] von mehr als 93.000 linearen Codes mit hoher Minimaldistanz über 24 verschiedenen endlichen Kettenringen aufgebaut. Mehr als 1.200 dieser Codes sind als optimal nachgewiesen. Außerdem wurden mehrere neue BTL- und BTKL-Codes gefunden. Einer von ihnen entpuppte sich als der erste Vertreter einer unendlichen Serie über $\mathbb{Z}_4$, für deren beide Anfangsglieder die BTL-Eigenschaft gezeigt werden konnte. Für einen anderen Code fand sich eine interessante geometrische Interpretation. Die Methoden wurden auch zur Konstruktion klassischer Codes über endlichen Körpern mit vorgeschriebener Automorphismengruppe eingesetzt. Dies führte zur Verbesserung der internationalen Tabellen für die beste bekannte Minimaldistanz an insgesamt 497 Stellen, wobei mindestens 38 der gefundenen Codes optimal sind.

Auf Grundlage dieser Ergebnisse ist festzustellen, dass die verallgemeinerte Version des Algorithmus sich als mächtiges Werkzeug für Konstruktionsprobleme der hier vorliegenden Art erwiesen hat. Die erzeugten Tabellen legen außerdem die Vermutung nahe, dass BTL- und BTKL-Codes eher „seltene“ Objekte sind, insbesondere für andere Kettenringe als $\mathbb{Z}_4$.

Zum Aufbau dieser Arbeit

Im ersten Kapitel fassen wir die wichtigsten Fakten und Definitionen über endliche Kettenringe zusammen und erläutern außerdem die Klasse der Galoisringe. Ferner finden sich hier Ausführungen über eine kanonische Darstellung dieser Ringe im Computer. Im zweiten Abschnitt geht es dann um die Theorie linearer Codes über Kettenringen, insbesondere werden Gewichtsfunktionen, die Grayabbildung und die Äquivalenz solcher Codes diskutiert. Abschnitt 2.4 stellt hierbei einen kleinen Exkurs dar und erklärt die Änderungen, die sich bei der Suche nach sogenannten *Arcs* ergeben.

Die Kapitel drei und vier widmen sich den heuristischen Methoden: Teil drei dreht sich um die direkte Übertragung der Ideen aus [Zwa07] von Codes über endlichen Körpern auf die Situation bei den Kettenringen. Kern dieses Kapitels ist die Definition einer heuristischen Bewertungsfunktion, auf der die nachfolgend erläuterten Algorithmen aufbauen. Außerdem geben wir einige Hinweise, wie diese effizient implementiert werden können. Das auf CD beigefügte Programm *Heurico* basiert auf den hier vorgestellten Methoden. Der vierte Abschnitt behandelt zunächst das Kramer-Mesner-Verfahren bei Codes über Körpern und beschreibt die notwendigen Anpassungen im Falle von Kettenringen. Anschließend nehmen wir eine Verallgemeinerung des heuristischen Ansatzes auf das Lösen der Diophantischen Ungleichungssysteme vor, die in dieser Situation auftreten. Da Systeme desselben Typs auch bei der Konstruktion anderer Objekte wie t-Designs oder Networkcodes vorkommen, ist der beschriebene Algorithmus auch dort ohne weitere Anpassungen einsetzbar. Er wird in dem auf der CD befindlichen Programm *Solver* umgesetzt.

Kapitel fünf zeigt die zum Aufbau der Codedatenbank durchgeführten Schritte; ein Schwerpunkt liegt dabei auf der Bestimmung der Automorphismengruppe eines endlichen Kettenringes. Der sechste Abschnitt stellt eine Dokumentation der Programme

Heurico und *Solver* dar. Neben Hinweisen zur Kompilierung erläutern wir hier auch die jeweils wichtigsten Aufrufparameter. Im letzten Kapitel geben wir dann eine Zusammenfassung der Suchergebnisse und erörtern kurz die interessantesten Entdeckungen.

Bemerkungen zur Notation

Bei allen in dieser Arbeit betrachteten Ringen R werden wir stillschweigend voraussetzen, dass sie ein Einselement 1_R besitzen, selbst wenn manche Definitionen oder Sätze eventuell auch ohne diese Annahme gültig wären. Bei Linksmoduln M über R schreiben wir gelegentlich zur Verdeutlichung auch ${}_RM$ und entsprechend M_R bei R-Rechtsmoduln. Für eine Menge X und $n \in \mathbb{N}^*$ bezeichnet X^n gleichermaßen die Menge der Zeilenvektoren wie die der Spaltenvektoren der Länge n über X. Welche Variante gemeint ist, wird stets aus dem Zusammenhang klar sein. Besitzt X ein (multiplikatives) Einselement, so ist e_i der i-te Einheitsvektor in X^n. Den Vektor aus lauter Einsen $\mathbb{1}$ interpretieren wir grundsätzlich als Spaltenvektor. Der Nullvektor wird symbolisiert durch 0 und manchmal auch durch $\vec{0}$, wenn betont werden soll, dass es sich um einen Vektor handelt. Für einen Zeilenvektor u und einen Spaltenvektor v über einem Ring R ist uv das Produkt der Vektoren im Sinne einer Matrixmultiplikation. Für die i-te Zeile einer Matrix A wird gelegentlich A_{i*} geschrieben werden. Ist S eine Menge mit Struktur und $U \subset S$, dann notiert $\langle U \rangle$ das Erzeugnis von U in S, also die kleinste Unterstruktur von S, die U umfasst. Das semidirekte Produkt zwischen zwei Gruppen G und H wird durch $G \rtimes H$ ausgedrückt; der zugehörige Homomorphismus von H in die Automorphismengruppe von G sollte immer aus dem Kontext ersichtlich sein. Operiert eine Gruppe G von rechts oder von links auf einer Menge X, so verwenden wir für die Menge der Bahnen dieser Operation einheitlich die Schreibweise $G\backslash\backslash X$. Ist H eine weitere Gruppe, dann bezeichnet $H \wr_X G$ das Kranzprodukt von H mit G. Mit $a = (a_0, a_1, \ldots, a_{l-1}) \vdash n$ ist gemeint, dass a eine Partition von n bildet, d. h. $\sum_{i=0}^{l-1} a_i = n$ und $a_i \geq a_{i+1}$, für $i < l - 1$, gilt. Zur Aufzählung der Elemente einer Multimenge Y benutzen wir doppelte geschweifte Klammern, etwa $Y = \{\{y_0, y_1, \ldots, y_{l-1}\}\}$. Das Enthaltensein einer Multimenge in einer anderen wird mit dem von normalen Mengen bekanntenen Symbolen $\subsetneq$ bzw. $\subset$ dargestellt. Zu guter Letzt sei noch auf das Symbolverzeichnis am Anfang dieser Arbeit verwiesen.

Danksagungen

An dieser Stelle möchte ich mich sehr herzlich bei Prof. Dr. Adalbert Kerber für die Betreuung dieser Arbeit bedanken; durch seine Initiative kam es, dass aus der spontan geborenen Idee für den heuristischen Algorithmus zunächst eine Diplomarbeit und nun sogar eine Dissertation wurde. Ganz besonders danke ich Michael Kiermaier für die gute Vorarbeit bei der Implementierung der Kettenringarithmetik, die gleichermaßen konstruktive wie unterhaltsame Zusammenarbeit beim Aufbau der Datenbank und sein jederzeit offenes Ohr bei Fragen aller Art. Ebenso gilt mein Dank Dr. Axel Kohnert für

den Einsatz von *Solver* im Rahmen der Suche nach linearen Codes über Körpern mit der Kramer-Mesner-Methode, was zu der enormen Zahl an Verbesserungen in diesem Bereich führte. Schließlich sei auch allen anderen gedankt, die zur Entstehung dieser Arbeit beigetragen haben, ohne dass sie hier im Einzelnen namentliche Erwähnung finden können.

Neubiberg, den 24. Juni 2011 Johannes Zwanzger

Kapitel 1.

Endliche Kettenringe

1.1. Grundlegende Eigenschaften und Definitionen

An dieser Stelle sollen zunächst die wichtigsten Definitionen und Sätze aus der Theorie der endlichen Kettenringe dargelegt werden, die für das Verständnis dieser Arbeit wichtig sind. Sie sind zumeist schon lange bekannt und finden sich in entsprechenden Lehrbüchern wie beispielsweise [McD74] oder [Lam01]. Wir folgen hier zu großen Teilen der Darstellung in [Kie06]. Dort finden sich auch Ausarbeitungen der Beweise zu vielen der hier zitierten Sätze.

Definition 1.1. Ein Ring R heißt *Linkskettenring*, wenn der Verband der Linksideale von R eine Kette bildet. Analog ist der Begriff *Rechtskettenring* definiert. Einen Ring, der gleichzeitig Links- und Rechtskettenring ist, bezeichnet man auch einfach als *Kettenring*.

Definition 1.2. Sei R ein Ring. Für eine Teilmenge $T \subset R$ sei (T) das von T erzeugte beidseitige Ideal in R. Sei nun $I \lhd R$ ein beidseitiges Ideal. Dann definieren wir die Potenzen von I wie folgt rekursiv: $I^0 := R$ und $I^{n+1} := (I \cdot I^n) := (\{i \cdot j : i \in I, j \in I^n\})$, für alle $n \in \mathbb{N}$.

Bemerkung 1.1. Da $1_R \in R$, gilt in der oben betrachteten Situation: $I^1 = I$. Für $n \geq 2$ ergibt einfaches Nachrechnen, dass $I^n = \{\sum_{j=0}^{l} i_{j,0} \cdot i_{j,1} \cdot \ldots \cdot i_{j,n-1} : l \in \mathbb{N},\, i_{j,k} \in I\}$.

Definition 1.3. Sei R ein Ring. Das *Jacobson-Radikal* von R ist definiert als der Schnitt aller maximalen Linksideale von R und wird mit $\mathrm{Rad}(R)$ bezeichnet.

Die erste Aussage des folgenden Satzes relativiert die scheinbar willkürliche Auszeichnung der Linksideale in obiger Definition:

Satz 1.1. *Für das Jacobson-Radikal eines Ringes R gilt:*

(i) *$Rad(R)$ ist identisch mit dem Schnitt der maximalen Rechtsideale von R.*

(ii) *$Rad(R)$ ist ein beidseitiges Ideal.*

(iii) *Ist R zusätzlich artinsch[1], so existiert ein $m \in \mathbb{N}^*$, so dass $Rad(R)^m = \{0\}$. Ist m minimal mit dieser Eigenschaft, so heißt m der* Nilpotenzindex *von $Rad(R)$.*

[1] Ein Ring R heißt *artinsch*, wenn jede absteigende Folge $R_0 \supset R_1 \supset R_2 \supset \ldots$ von Untermoduln von ${}_RR$ bzw. R_R nach endlichen vielen Gliedern stationär wird. Insbesondere sind also alle endlichen Ringe artinsch.

Bevor wir einen sehr wichtigen Satz über die Struktur endlicher Kettenringe anführen, benötigen wir noch den darin auftretenden Begriff des lokalen Ringes.

Definition 1.4. Ein Ring R heißt *lokaler Ring*, wenn in ihm genau ein maximales Linksideal existiert.

Bemerkung 1.2. Ist R ein lokaler Ring, so ist sein maximales Linksideal natürlich gerade das Jacobson-Radikal $\mathrm{Rad}(R)$. Weiterhin ändert sich auch hier die Definition nicht, wenn man das Wort Linksideal durch Rechtsideal ersetzt, das heißt es gilt:

Satz 1.2. *R lokaler Ring $\Leftrightarrow$ $Rad(R)$ ist das einzige maximale Linksideal von R $\Leftrightarrow$ $Rad(R)$ ist das einzige maximale Rechtsideal von R.*

Nun zu dem bereits angekündigten Struktursatz. Er entstammt im Wesentlichen [CD73]. Die hier zitierte, leicht modifizierte Version ist [Kie06] entnommen. Dort findet sich auch ein ausführlicher Beweis.

Satz 1.3. *Sei R ein endlicher Ring, $\theta \in Rad(R)$ und m der nach Satz 1.1, Punkt (iii) wohldefinierte Nilpotenzindex von $Rad(R)$. Falls $m > 1$, so gelte weiter $\theta \notin Rad(R)^2$. Dann sind äquivalent:*

(i) *R ist ein Linkskettenring.*

(ii) *R ist ein Rechtskettenring.*

(iii) *R ist ein lokaler Ring mit $Rad(R) = R\theta$.*

(iv) *$R/Rad(R)$ ist isomorph zu einem endlichen Körper $\mathbb{F}_q$ mit einer Primzahlpotenz q. Die einzigen Links- und Rechtsideale von R sind gerade die Potenzen $Rad(R)^k$, $k = 0, \ldots, m$. Diese sind allesamt beidseitige Hauptideale und es gilt:*

$$Rad(R)^k = R\theta^k = \theta^k R \text{ mit } \left|Rad(R)^k\right| = q^{m-k} \ (k = 0, \ldots, m).$$

Weiterhin lassen sich die stufenweisen Differenzen der Idealkette wie folgt mit Hilfe der Einheitengruppe $R^\times$ darstellen:

$$Rad(R)^k \setminus Rad(R)^{k+1} = R^\times\theta^k = \theta^k R^\times \ (k \in \{0, 1, \ldots, m-1\}).$$

Insbesondere ist also $R^\times = R \setminus Rad(R)$.

Definition 1.5. Sei R ein endlicher Kettenring.

(i) Der Nilpotenzindex von $\mathrm{Rad}(R)$ wird auch die *Kettenlänge* von R genannt; wir werden ihn in dieser Arbeit mit m_R abkürzen.

(ii) Weiter definieren wir die Werte q_R, $r_R \in \mathbb{N}^*$ sowie $p_R \in \mathbb{P}$ durch $R/Rad(R) \cong \mathbb{F}_{q_R}$ und $q_R = p_R^{r_R}$.

(iii) Im Weiteren sei θ_R stets ein beliebig, aber fest gewählter Erzeuger von $\mathrm{Rad}(R)$.

(iv) Für ein Element $x \in R$ sei $h \in \mathbb{N}$ maximal mit $h \leq m$ und $x \in \mathrm{Rad}(R)^h$. Dann heißt h die *Höhe* von x, welche im weiteren Verlauf dieser Arbeit mit $\mathfrak{h}(x)$ notiert wird. Die kleinste Zahl $g \in \mathbb{N}$ mit $x\theta_R^g = 0$ wird als die *Periode* von x bezeichnet. Es gilt: $h + g = m$.

Beispiel 1.1.

(i) Für eine beliebige Primzahl p und $n \in \mathbb{N}^*$ ist der Ring $R := \mathbb{Z}_{p^n}$ ein endlicher Kettenring. Es gilt: $q_R = p$ und $m_R = n$. Außerdem kann $\theta_R := \overline{p}$ gewählt werden. Die Idealkette von R lautet $R \supsetneq (\overline{p}) \supsetneq (\overline{p^2}) \supsetneq \ldots \supsetneq (\overline{p^{n-1}}) \supsetneq \{\overline{0}\}$.

(ii) Hat dagegen eine natürliche Zahl s mindestens zwei verschiedene Primteiler p_1 und p_2, so ist $S := \mathbb{Z}_s$ kein Kettenring, denn von den beiden Idealen $(\overline{p_1})$ und $(\overline{p_2})$ ist keines vollständig im anderen enthalten.

(iii) Für eine Primzahlpotenz $q = p^r$ ist der endliche Körper $F := \mathbb{F}_q$ ein Kettenring mit $\theta_F = 0$, Kettenlänge $m_F = 1$ und den weiteren Parametern $q_F = q$, $p_F = p$, $r_F = r$.

Bemerkung 1.3. In den folgenden Abschnitten werden wir, sofern keine Verwechslungsgefahr besteht, die Parameter q_R, p_R, r_R, m_R und θ_R eines endlichen Kettenrings R zur leichteren Lesbarkeit mit q, p, r, m und θ abkürzen.

Lemma 1.4. *Für $r, s \in R$ mit $\mathfrak{h}(r) < \mathfrak{h}(s)$ gilt $\mathfrak{h}(r + s) = \mathfrak{h}(r)$.*

Beweis. Sei $t := r + s$ und seien $h_r = \mathfrak{h}(r)$, $h_s = \mathfrak{h}(s)$ und $h_t = \mathfrak{h}(t)$ die zugehörigen Höhen. Da $\mathrm{Rad}(R)^{\mathfrak{h}(r)}$ additiv abgeschlossen ist, gilt $t \in \mathrm{Rad}(R)^{\mathfrak{h}(r)}$ und $h_t \geq h_r$. Wäre allerdings $h_t > h_r$, so hätten wir $\mu := \min(h_s, h_t) > h_r$ und damit $-s, t \in \mathrm{Rad}(R)^\mu$, aber $r = t - s \notin \mathrm{Rad}(R)^\mu$. $\mathrm{Rad}(R)^\mu$ wäre also nicht additiv abgeschlossen, ein Widerspruch. Folglich muss $h_t = h_r$ gelten.

□

1.2. Teichmüllermengen, θ-adische Entwicklung

Jetzt wollen wir auf eine wichtige Möglichkeit zur Darstellung der Elemente eines endlichen Kettenringes eingehen:

Definition 1.6. Sei R ein endlicher Kettenring. $\mathcal{A} \subset R$ heißt eine *Teichmüllermenge* von R, wenn $0 \in \mathcal{A}$ und $\mathcal{A}^* := \mathcal{A} \setminus \{0\}$ ein multiplikativ abgeschlossenes Vertretersystem von $(R/\mathrm{Rad}(R))^\times$ ist. $\mathcal{A}^*$ ist isomorph zur multiplikativen Gruppe des endlichen Körpers $\mathbb{F}_q$ und daher eine zyklische Untergruppe von $R^\times$. Wir nennen $\mathcal{A}^*$ dann eine *Teichmüllergruppe* von R und ein sie erzeugendes Element α einen *Teichmüllererzeuger* von R. In diesem Zusammenhang sind die folgenden Sätze sehr wichtig (vgl. [Kie06, Abschnitt 4.2, 4.3 und 4.5]):

Satz 1.5. *Sei R ein endlicher Kettenring. Dann gilt:*

(i) *R besitzt mindestens eine Teichmüllermenge.*

(ii) *Die Teichmüllergruppen von R sind gerade die zyklischen Untergruppen der Ordnung $q-1$ von $R^\times$.*

(iii) *Alle Teichmüllergruppen von R sind in $R^\times$ zueinander konjugiert. Insbesondere gibt es nur eine einzige Teichmüllergruppe, wenn R kommutativ ist.*

Satz 1.6. *Sei R ein endlicher Kettenring und $\mathcal{A}$ eine Teichmüllermenge von R. Dann hat jedes Element $r \in R$ eine* eindeutige *Darstellung der Form*

$$r = \sum_{i=0}^{m-1} \alpha_i \theta^i$$

mit $\alpha_i \in \mathcal{A}$ für $i \in \{0, 1, \ldots, m-1\}$. Für den Fall $\theta = 0$ verwenden wir hierbei die Konvention $0^0 = 1$. Zusätzlich gilt: $r \in Rad(R)^k \Leftrightarrow \forall i \in \{0, 1, \ldots, k-1\} : \alpha_i = 0$.

Definition 1.7. Die Darstellung in 1.6 heißt die *rechts-θ-adische Entwicklung* von r nach der Teichmüllermenge $\mathcal{A}$. Wir werden im Folgenden aber vereinfachend nur von der θ-adischen Entwicklung von r sprechen.

Satz 1.7. *Die folgenden Aussagen sind für einen Kettenring R mit Teichmüllermenge $\mathcal{A}$ äquivalent:*

(i) *$\mathcal{A}$ ist additiv abgeschlossen.*

(ii) *$\mathcal{A} \cong \mathbb{F}_q$.*

(iii) *$char(R) = p$.*

Beispiel 1.2. Sei $R := \mathbb{Z}_9$. Dann ist $\mathcal{A} := \{\overline{0}, \overline{1}, \overline{8}\}$ die einzige Teichmüllermenge von R und $\alpha = \overline{8}$ der in diesem Fall eindeutige Teichmüllererzeuger von $\mathcal{A}^*$. Mit $\theta := \overline{3}$ ergibt sich die folgende θ-adische Entwicklung der Elemente von R:

r	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$	$\overline{8}$
(α_0, α_1)	$(\overline{0}, \overline{0})$	$(\overline{1}, \overline{0})$	$(\overline{8}, \overline{1})$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{1})$	$(\overline{8}, \overline{8})$	$(\overline{0}, \overline{8})$	$(\overline{1}, \overline{8})$	$(\overline{8}, \overline{0})$

Bemerkung 1.4. Sei R ein endlicher Kettenring mit Teichmüllermenge $\mathcal{A}$ und dem zugehörigen Teichmüllererzeuger α.

(i) Die θ-adische Entwicklung ist in der Regel nicht additiv: Für Ringelemente $r, s \in R$ mit den θ-adischen Entwicklungen $r = \sum_{i=0}^{m-1} \alpha_i \theta^i$ und $s = \sum_{i=0}^{m-1} \beta_i \theta^i$ gilt natürlich $r+s = \sum_{i=0}^{m-1} (\alpha_i + \beta_i)\theta^i$, dies ist jedoch normalerweise nicht die θ-adische Entwicklung von $r+s$, da ggf. $\alpha_i + \beta_i \notin \mathcal{A}$.

(ii) Die Berechnung der θ-adischen Entwicklung der Ringelemente kann rekursiv erfolgen:

(a) Starte mit $\mathrm{Rad}(R)^m = \{0\}$: Die Entwicklung der Null ist stets $0 = \sum_{i=0}^{m-1} 0 \cdot \theta^i$.

(b) Ist $i \geq 1$ und die Entwicklung für die Menge $\mathrm{Rad}(R)^i$ bereits bekannt, dann kann sie für alle Elemente $r \in \mathrm{Rad}(R)^{i-1} \setminus \mathrm{Rad}(R)^i = R^\times \theta^{i-1}$ so ermittelt werden: Schreibe r in der Form $r = e\theta^{i-1}$ mit einer Einheit e. Da $\mathcal{A}$ ein Vertretersystem von $R/\mathrm{Rad}(R)$ bildet, ist e darstellbar als $e = \alpha_e + r_e\theta$ mit $\alpha_e \in \mathcal{A}$, $r_e \in R$. Es folgt $r = \alpha_e\theta^{i-1} + r_e\theta^i$. Die Entwicklung von $r_e\theta^i$ ist nach Voraussetzung schon bekannt und ihre Koeffizienten verschwinden nach Satz 1.6 für alle θ-Potenzen unterhalb von i. Daher ist die θ-adische Entwicklung von r gerade die von $r_e\theta^i$ mit dem zusätzlichen Summanden $\alpha_e\theta^{i-1}$.

(iii) Die Elemente von R können in einem Computerprogramm mittels

$$\psi : \mathcal{A} \to \{0, 1, \ldots, q-1\}, \begin{cases} 0 & \mapsto 0 \\ \alpha^i & \mapsto i+1 \end{cases}$$

wie folgt dargestellt werden:

$$\phi : R \to \{0, 1, \ldots, q^m - 1\}, \sum_{i=0}^{m-1} \alpha_i \theta^i \mapsto \sum_{i=0}^{m-1} \psi(\alpha_i) q^i.$$

Dann ist beispielsweise die Multiplikation eines Ringelementes mit θ von rechts einfach (also ohne Nachschlagen in einer im Speicher abgelegten Verknüpfungstafel) möglich, denn $\phi(r \cdot \theta) \equiv \phi(r) \cdot q \mod q^m$.

Definition 1.8. Für zwei endliche Kettenringe R und S mit $q_R = q_S$ und $m_R = m_S$ seien $\mathcal{A}$ eine Teichmüllermenge von R mit Teichmüllererzeuger α, β ein Teichmüllererzeuger in S sowie θ und ω Erzeuger der Ideale $\mathrm{Rad}(R)$ bzw. $\mathrm{Rad}(S)$. Durch die Festlegungen $\chi_\beta^\omega(0) := 0$ und $\chi_\beta^\omega(\alpha^j) := \beta^j$ ist die Abbildung χ_β^ω auf ganz $\mathcal{A}$ erklärt und kann, für ein allgemeines Ringelement $r \in R$ mit der Teichmüllerentwicklung $r = \sum_{i=0}^{m-1} \alpha_i \theta^i$, fortgesetzt werden durch

$$\chi_\beta^\omega(r) := \sum_{i=0}^{m-1} \chi_\beta^\omega(\alpha_i) \omega^i.$$

Bemerkung 1.5. Für einen Ringisomorphismus $\sigma : R \to S$ stimmt σ mit $\chi_{\sigma(\alpha)}^{\sigma(\theta)}$ überein. σ ist also schon durch die Werte $\sigma(\alpha)$ und $\sigma(\theta)$ festgelegt. Umgekehrt ist jedes χ_β^ω zwar eine Bijektion, aber nicht unbedingt ein Ringisomorphismus.

1.3. Galoisringe

Jetzt soll eine spezielle Klasse von endlichen Ringen, die sogenannten *Galoisringe*, näher betrachtet werden; erstmals dürfte dies in [Kru24], dort noch unter der Bezeichnung „Grundringe“, geschehen sein. Wir werden sehen, dass jeder Galoisring ein Kettenring

ist. Vor allem aber lassen sich alle Kettenringe auf relativ einfache Weise als Restklassenring eines Schiefpolynomrings über einem Galoisring darstellen. Für eine umfassende Einführung zu dem Thema verweisen wir auf [McD74]; unsere Ausführungen orientieren sich an [Nec91].

Definition 1.9. Ein endlicher, kommutativer Ring R heißt *Galoisring*, wenn die Menge der Nullteiler von R gerade durch die Menge $\overline{p}R \setminus \{0\}$ gegeben ist mit einer Primzahl p, wobei wir $\overline{p}$ als Element von R auffassen vermöge $\overline{p} := \underbrace{1_R + 1_R + \cdots + 1_R}_{p \text{ mal}}$.

Beispiel 1.3. Sei p eine Primzahl.

(i) $R := \mathbb{Z}_{p^n}$ ist ein Galoisring mit Nullteilermenge $\overline{p}R \setminus \{0\} = \{\overline{p \cdot m} : 1 \leq m < p^{n-1}\}$.

(ii) Für $q = p^r$ ist $\mathbb{F}_q$ ein Galoisring, denn $\mathbb{F}_q$ besitzt keine Nullteiler und $\overline{p}\mathbb{F}_q \setminus \{0\} = \emptyset$.

Der folgende Satz stellt unter anderem klar, dass es sich bei den Galoisringen um Kettenringe handelt:

Satz 1.8. *Sei R ein Galoisring und $N := \overline{p}R$. Dann gilt:*

(i) $R^\times = R \setminus N$.

(ii) *R ist ein lokaler Ring und sein maximales Ideal ist N. Weiter ist $R/N \cong \mathbb{F}_q$, wobei $q = p^r$ für ein geeignetes $r \in \mathbb{N}^*$.*

(iii) *Die Charakteristik von R ist eine Potenz von p, also char(R)= p^m für $m \in \mathbb{N}^*$ geeignet.*

(iv) *Die Ideale von R sind genau die Glieder der Kette $R = N^0 \supsetneq N^1 = (\overline{p})^1 R \supsetneq \ldots \supsetneq N^{m-1} = (\overline{p})^{m-1} R \supsetneq N^m = (\overline{p})^m R = \{0\}$. R ist also ein Kettenring mit Kettenlänge $m_R = m$ und $p_R = p$, $r_R = r$, $q_R = q$. Ferner kann $\theta := \overline{p}$ gewählt werden.*

Definition 1.10. Mit den Bezeichnern von oben sei, für $r \in R$, $\overline{r} := r + N$ die Projektion von r auf $\overline{R} := R/N \cong \mathbb{F}_q$. Diese induziert auf natürliche Weise einen Epimorphismus von $R[X]$ auf $\overline{R}[X]$ mittels $f = \sum_{i=0}^{j} r_i X^i \mapsto \overline{f} := \sum_{i=0}^{j} \overline{r}_i X^i$. Ein normiertes Polynom $f \in R[X]$ heißt nun *Galoispolynom*, wenn $\overline{f}$ in $\overline{R}[X]$ irreduzibel ist.

Satz 1.9. *Sei R ein Galoisring mit Charakteristik p^m und Ordnung q^m. Sei f ein Galoispolynom vom Grad n. Dann ist der Ring $S := R[X]/(f)$ wieder ein Galoisring mit char(S)= p^m und $|S| = (q^n)^m = q^{mn}$, das heißt $m_S = m$, $p_S = p$, $r_S = n \cdot r$, $q_S = q^n$.*

Definition 1.11. Da der Ring R in obigem Satz ein Unterring von S ist, nennen wir S eine *Galoiserweiterung* von R vom Grad n.

Der nächste Satz zeigt eine starke Analogie zwischen den Galoisringen und den endlichen Körpern:

Satz 1.10. *Sei S eine Galoiserweiterung des Galoisringes R vom Grad n. Sei f ein Galoispolynom vom Grad k über R. Dann gilt:*

(i) f besitzt Wurzeln in S genau dann, wenn $k \mid n$.

(ii) Wenn $k \mid n$, so liegen in S sogar genau k Wurzeln $s_0, \ldots, s_{k-1}$ von f. Diese sind paarweise verschieden modulo $\overline{p}S$ und es gilt: $f(x) = (x - s_0) \cdots (x - s_{k-1})$.

(iii) Für jedes $s \in S$ gilt $S = R[s]$ genau dann, wenn s die Wurzel eine Galoispolynoms vom Grad n über R ist.

Satz 1.11. *Sei S ein Galoisring der Charakteristik p^m und Ordnung p^{rm}. Sei weiter f ein beliebiges Galoispolynom vom Grad r in $\mathbb{Z}_{p^m}[X]$. Dann ist S isomorph zu $\mathbb{Z}_{p^m}[X]/(f)$.*

Korollar 1.12. *Zwei Galoisringe derselben Charakteristik und Ordnung sind zueinander isomorph.*

Bemerkung 1.6. Aufgrund von Korollar 1.12 ist es zulässig, von *dem* Galoisring mit Charakteristik p^m und Ordnung p^{rm} zu sprechen. Wir werden uns im Folgenden mit der Notation $\mathrm{GR}(p^{rm}, p^m)$ auf ihn beziehen.

Zur standardisierten Darstellung von $\mathrm{GR}(p^{rm}, p^m)$ im Computer als $\mathbb{Z}_{p^m}[X]/(f)$ ist es wünschenswert, eine kanonische Form für das Galoispolynom f zu verwenden. Im Falle der endlichen Körper $\mathbb{F}_{p^r}$ wurden hierfür von Richard Parker die rekursiv definierten *Conwaypolynome* eingeführt[2]:

Definition 1.12. Das Conwaypolynom $f^C_{p,r}$ ist das lexikographisch kleinste[3] irreduzible Polynom vom Grad r in $\mathbb{F}_p[X]$ mit folgenden Eigenschaften:

(i) $f^C_{p,r}$ ist normiert.

(ii) $f^C_{p,r}$ ist primitiv.

(iii) Für jeden Teiler t von r gilt, mit $s := \frac{p^r-1}{p^t-1}$: $f^C_{p,r} \mid f^C_{p,t}(X^s)$, das heißt die s-te Potenz einer Wurzel von $f^C_{p,r}$ ist eine Wurzel von $f^C_{p,t}$.

Eine umfassende Tabelle über die bisher berechneten Conwaypolynome ist online abrufbar unter [Lüb]; die meisten davon sind auch in Computeralgebrasystemen wie *GAP* [GAP] oder *MAGMA* [Com] integriert. Das Konzept der Conwaypolynome lässt sich von den Körpern auf die Galoisringe übertragen, indem man $f^C_{p,r}$ von $\mathbb{Z}_p$ nach $\mathbb{Z}_{p^m}$ „liftet“.[4] Zum Beweis benötigen wir den folgenden Satz, der eine Konsequenz aus dem Henselschen Lemma ist:

[2] In [Sch92] wird diesbezüglich auf einen Vortrag von Parker in Heidelberg verwiesen [Par90].

[3] Hinsichtlich der Ordnung: $a_r X^r + a_{r-1} X^{r-1} + \cdots + a_1 X^1 + a_0 < b_r X^r + b_{r-1} X^{r-1} + \cdots + b_1 X^1 + b_0 :\Leftrightarrow \exists i : a_j = b_j \forall j > i \wedge (-1)^{r-i} a_i < (-1)^{r-i} b_i$, wobei $\overline{0} < \overline{1} < \cdots < \overline{p-1}$ in $\mathbb{F}_p$.

[4] An dieser Idee sowie ihrer Umsetzung war Michael Kiermaier maßgeblich beteiligt.

Satz 1.13. *Sei f ein normiertes Polynom aus $\mathbb{Z}_{p^m}[X]$, so dass $\overline{f}$ über $\mathbb{Z}_p$ die Zerlegung $\overline{f} = g_0 g_1 \cdots g_{s-1}$ in die normierten, paarweise teilerfremden[5] Polynome $g_0, g_1, \ldots, g_{s-1}$ besitzt. Dann gibt es normierte, paarweise teilerfremde Polynome $f_0, f_1, \ldots, f_{s-1}$ über $\mathbb{Z}_{p^m}$, so dass gilt: $f = f_0 f_1 \cdots f_{s-1}$ in $\mathbb{Z}_{p^m}[X]$ und $\overline{f_i} = g_i$ für $i \in \{0, 1, \ldots, s-1\}$. Die f_i sind dabei eindeutig bestimmt.*

Damit können wir die Conwaypolynome wie gewünscht verallgemeinern:

Satz 1.14. *Für jedes $p \in \mathbb{P}$ und jedes $m \in \mathbb{N}^*$ existiert zum Conwaypolynom $f^C_{p,r}$ ein eindeutig bestimmtes Polynom $f^C_{p,r,m} \in \mathbb{Z}_{p^m}[X]$ mit folgenden Eigenschaften:*

(i) $f^C_{p,r,m} \equiv f^C_{p,r} \mod p$.

(ii) $f^C_{p,r,m}$ *ist normiert.*

(iii) $f^C_{p,r,m} \mid X^{p^r-1} - 1$.

Weiterhin gilt dann:

(a) $f^C_{p,r,m} \equiv f^C_{p,r,m'} \mod p^{m'}$ *für* $m' \leq m$.

(b) $X + (f^C_{p,r,m})$ *ist ein Teichmüllererzeuger von* $\mathbb{Z}_{p^m}[X]/(f^C_{p,r,m}) \cong GR(p^{rm}, p^m)$.

(c) Für jeden Teiler t von r gilt, mit $s := \frac{p^r-1}{p^t-1}$: $f^C_{p,r,m} \mid f^C_{p,t,m}(X^s)$.

Beweis. Zunächst zu Existenz und Eindeutigkeit der Polynome $f^C_{p,r,m}$: $f^C_{p,r}$ teilt $X^{p^r-1}-1$ in $\mathbb{Z}_p[X]$ und die Wurzeln von $X^{p^r-1} - 1$ sind einfach, folglich gibt es eine Zerlegung $X^{p^r-1} - 1 = g_0 g_1 \cdots g_{l-1}$ mit $g_0 := f^C_{p,r}$, bei der die g_i paarweise teilerfremd und normiert sind. Nach Satz 1.13 gibt es dann eindeutig bestimmte, paarweise teilerfremde und normierte Polynome $f_0, f_1 \ldots, f_{l-1} \in \mathbb{Z}_{p^m}[X]$ mit $\overline{f_i} = g_i$ und $X^{p^r-1} - 1 = f_0 f_1 \cdots f_{l-1}$. Dann besitzt $f^C_{p,r,m} := f_0$ die Eigenschaften (i), (ii) und (iii). Die Annahme eines weiteren solchen Polynoms $h_0 \neq f_0$ ergäbe aufgrund der Existenz einer Zerlegung von $\frac{X^{p^r-1}-1}{h_0} = h_1 h_2 \cdots h_{l-1}$ mit $\overline{h_i} = g_i$ in paarweise teilerfremde, normierte Polynome über $\mathbb{Z}_{p^m}[X]$ eine zweite Zerlegung $X^{p^r-1} - 1 = h_0 h_1 \cdots h_{l-1}$, also einen Widerspruch zu Satz 1.13.

Nun zu den weiteren Folgerungen: Da $f^C_{p,r,m} \mod p^{m'}$ ganz offensichtlich die Eigenschaften (i), (ii) und (iii) für $m' \leq m$ erfüllt und diese $f^C_{p,r,m'}$ eindeutig festlegen, ist (a) klar. Wegen der Primitivität von $f^C_{p,r}$ gilt $\operatorname{ord}(X + (f^C_{p,r})) = p^r - 1$ in $\mathbb{Z}_p[X]/(f^C_{p,r})$, also $\operatorname{ord}(X + (f^C_{p,r,m})) \geq p^r - 1$ in $\mathbb{Z}_{p^m}[X]/(f^C_{p,r,m})$. Wegen Eigenschaft (iii) muss aber auch $\operatorname{ord}(X + (f^C_{p,r,m})) \leq p^r - 1$ gelten, also $\operatorname{ord}(X + (f^C_{p,r,m})) = p^r - 1$. Nach Satz 1.5 folgt damit Punkt (b). Es bleibt noch Eigenschaft (c) zu zeigen: Wie zuvor $X^{p^r-1} - 1$ besitzt auch $X^{p^t-1} - 1$ in $\mathbb{Z}_{p^m}[X]$ eine Zerlegung $X^{p^t-1} - 1 = k_0 k_1 \cdots k_{u-1}$ mit $k_0 = f^C_{p,t,m}$ und paarweise teilerfremden, normierten Polynomen k_i. In $\mathbb{Z}_p[X]/(f^C_{p,r})$ gilt: $\overline{k_0}(X^s + (f^C_{p,r})) = f^C_{p,t}(X^s + (f^C_{p,r})) = 0 + (f^C_{p,r})$ und $\overline{k_i}(X^s + (f^C_{p,r})) \neq 0 + (f^C_{p,r})$ für $i \neq 0$. Es folgt, dass

[5] Polynome f_1 und f_2 aus $\mathbb{Z}_{p^m}[X]$ heißen *teilerfremd*, wenn es $\lambda, \mu \in \mathbb{Z}_{p^m}$ gibt mit $\lambda f_1 + \mu f_2 = 1$.

$k_i(X^s + (f^C_{p,r,m}))$ für $i > 0$ eine Einheit in $\mathbb{Z}_{p^m}[X]/(f^C_{p,r,m})$ ist. Da aber für X^s als Wurzel von $X^{p^t-1} - 1$ die Gleichung

$$k_0(X^s + (f^C_{p,r,m}))k_1(X^s + (f^C_{p,r,m})) \cdots k_{u-1}(X^s + (f^C_{p,r,m})) = 0 + (f^C_{p,r,m})$$

gilt, muss bereits $k_0(X^s+(f^C_{p,r,m})) = f^C_{p,t,m}(X^s+(f^C_{p,r,m})) = 0+(f^C_{p,r,m})$ sein. Das entspricht genau der Behauptung in Punkt (c).

□

Für die Darstellung des Galoisrings $\mathrm{GR}(p^{rm}, p^m)$ ist also mit $f^C_{p,r,m}$ ein Polynom in $\mathbb{Z}_{p^m}[X]$ ausgezeichnet, welches im Fall $m = 1$ mit dem klassischen Conwaypolynom $f^C_{p,r}$ zusammenfällt. Das begründet folgende Definition:

Definition 1.13. Die Polynome $f^C_{p,r,m}$ aus dem vorangehenden Satz nennen wir die *verallgemeinerten Conwaypolynome vom Grad r über $\mathbb{Z}_{p^m}$.*

Bemerkung 1.7. Die Berechnung des verallgemeinerten Conwaypolynoms $f^C_{p,r,m}$ kann man wie folgt durchführen: $\overline{\beta} := X+(f^C_{p,r})$ ist ein zyklischer Erzeuger der multiplikativen Gruppe von $\mathbb{Z}_p[X]/(f^C_{p,r}) \cong \mathbb{F}_{p^r}$. Insbesondere ist $\mathrm{ord}(\overline{\beta}) = q - 1$. Nun wähle $g \in \mathbb{Z}_{p^m}[X]$ mit $g = f^C_{p,r} \mod p$; am einfachsten interpretiert man hierzu die Koeffizienten von $f^C_{p,r}$ als Elemente von $\mathbb{Z}_{p^m}$. Sei $R := \mathbb{Z}_{p^m}[X]/(g)$ und $\beta := X + (g) \in R$. Dann gilt $(q-1) \mid \mathrm{ord}(\beta)$. g ist ein Galoispolynom vom Grad r und daher $R \cong \mathrm{GR}(p^{rm}, p^m)$. Für die Einheitengruppe gilt $|R^\times| = (q-1)q^{m-1}$. Für β als Einheit in R ergibt sich damit insgesamt, dass $\mathrm{ord}(\beta) = (q-1)p^t$ mit $0 \le t \le r(m-1)$. Dann hat das Element $\gamma := \beta^{(p^t)}$ gerade die Ordnung $q-1$, also ist γ ein Teichmüllererzeuger von R. Als Bild einer Potenz des Frobeniusautomorphismus $X \mapsto X^p$ von $\overline{\beta}$ ist $\overline{\gamma}$ außerdem ebenfalls eine Wurzel von $f^C_{p,r}$. Mit dem Ansatz

$$-\gamma^r = \sum_{i=0}^{r-1} c_i \gamma^i,\ c_i \in \mathbb{Z}_{p^m}$$

bestimmen wir nun ein normiertes Polynom $f := \sum_{i=0}^{r-1} c_i X^i + X^r$ vom Grad r, so dass $f(\gamma) = 0$ in R. Für ein solches f gilt, da $\overline{f}(\overline{\gamma}) = 0$, $f \equiv f^C_{p,r} \mod p$ und außerdem $f \mid X^{p^r-1} - 1$, also ist f im Falle der Lösbarkeit der obigen Gleichung eindeutig bestimmt und identisch mit $f^C_{p,r,m}$. Dass eine Lösung f existiert, sieht man daran, dass es kein von Null verschiedenes Polynom h vom Grad kleiner r mit γ als Wurzel geben kann: Die Betrachtung dieser Situation modulo p lieferte $\overline{h} \mid f^C_{p,r}$, einen Widerspruch zur Irreduzibilität von $f^C_{p,r}$, womit sich die Existenz von f aus der nun offensichtlichen Beobachtung ergibt, dass $-\gamma^r \in \{\sum_{i=0}^{r-1} d_i\gamma^i : d_i \in \mathbb{Z}_{p^m}\} = R$, denn die aufgelisteten Elemente sind alle verschieden und $|R| = (p^m)^r$.

Anmerkung: Die Gleichung $-\gamma^r = \sum_{i=0}^{r-1} c_i\gamma^i$ lässt sich als lineares Gleichungssystem $Ax = b$ über $\mathbb{Z}_{p^m}$ behandeln. Dabei ist die quadratische Matrix $A = (a_{ij})_{i,j=0..r-1}$ definiert durch die Gleichung $\gamma^j = \sum_{i=0}^{r-1} a_{ij}X^i + (g)$ und der Vektor $b = (b_0, \ldots, b_{r-1})^T$ mittels $-\gamma^r = \sum_{i=0}^{r-1} b_i X^i + (g)$. Tabelle 1.1 zeigt eine Übersicht über die verallgemeinerten Conwaypolynome für kleine Parameter.

Tabelle 1.1.: Verallgemeinerte Conwaypolynome

p	r	m	$f^C_{p,r,m} \in \mathbb{Z}_{p^m}[X]$
2	2	16	$X^2 + X + 1$
2	3	16	$X^3 + 40870X^2 + 40869X + 65535$
2	4	16	$X^4 + 8812X^3 + 65534X^2 + 56723X + 1$
2	5	16	$X^5 + 33032X^4 + 51364X^3 + 49503X^2 + 31170X + 65535$
2	6	16	$X^6 + 61926X^5 + 58949X^4 + 28781X^3 + 2756X^2 + 36535X + 1$
3	2	10	$X^2 + 56354X + 59048$
3	3	10	$X^3 + 12999X^2 + 21134X + 1$
3	4	10	$X^4 + 43964X^3 + 23208X^2 + 51783X + 59048$
3	5	10	$X^5 + 33867X^4 + 7290X^3 + 30729X^2 + 48230X + 1$
3	6	10	$X^6 + 46224X^5 + 17027X^4 + 13896X^3 + 28168X^2 + 18563X + 59048$
5	2	6	$X^2 + 7839X + 14557$
5	3	6	$X^3 + 8530X^2 + 7623X + 1068$
5	4	6	$X^4 + 345X^3 + 7004X^2 + 15034X + 14557$
5	5	6	$X^5 + 14290X^4 + 8490X^3 + 2895X^2 + 13904X + 1068$
5	6	6	$X^6 + 15430X^5 + 4536X^4 + 11164X^3 + 7271X^2 + 1265X + 14557$
7	2	5	$X^2 + 1945X + 1354$
7	3	5	$X^3 + 1630X^2 + 15414X + 15453$
7	4	5	$X^4 + 7035X^3 + 8790X^2 + 11554X + 1354$
7	5	5	$X^5 + 1813X^4 + 1533X^3 + 8407X^2 + 9486X + 15453$
7	6	5	$X^6 + 3332X^5 + 9199X^4 + 10414X^3 + 977X^2 + 7951X + 1354$
11	2	4	$X^2 + 227X + 12575$
11	3	4	$X^3 + 1155X^2 + 7647X + 2066$
11	4	4	$X^4 + 352X^3 + 10183X^2 + 2254X + 12575$
11	5	4	$X^5 + 9779X^4 + 5214X^3 + 6863X^2 + 5731X + 2066$
11	6	4	$X^6 + 10065X^5 + 7978X^4 + 7605X^3 + 11941X^2 + 13405X + 12575$
13	2	4	$X^2 + 25869X + 4812$
13	3	4	$X^3 + 4563X^2 + 6918X + 23749$
13	4	4	$X^4 + 15314X^3 + 3305X^2 + 23932X + 4812$
13	5	4	$X^5 + 19188X^4 + 25870X^3 + 28275X^2 + 6114X + 23749$
13	6	4	$X^6 + 27404X^5 + 13715X^4 + 3689X^3 + 14649X^2 + 1480X + 4812$
17	2	3	$X^2 + 4878X + 802$
17	3	3	$X^3 + 2754X^2 + 3163X + 4111$
17	4	3	$X^4 + 1598X^3 + 1639X^2 + 1642X + 802$
17	5	3	$X^5 + 918X^4 + 2754X^3 + 3672X^2 + 1735X + 4111$
17	6	3	$X^6 + 289X^5 + 4541X^4 + 340X^3 + 4379X^2 + 2893X + 802$
19	2	3	$X^2 + 5129X + 5170$
19	3	3	$X^3 + 6764X^2 + 2284X + 1689$
19	4	3	$X^4 + 6783X^3 + 3460X^2 + 562X + 5170$
19	5	3	$X^5 + 760X^4 + 2071X^3 + 3572X^2 + 4812X + 1689$
19	6	3	$X^6 + 3990X^5 + 3933X^4 + 4938X^3 + 1651X^2 + 1469X + 5170$

Beispiel 1.4. Sei $p = 2$, $r = 4$, $m = 3$ und $t = 2$. Aus Tabelle 1.1 entnehmen wir (durch Reduktion der Koeffizienten von $f^C_{2,4,16}$ modulo 8), dass $f^C_{2,4,3} = X^4+4X^3+6X^2+3X+1$. Analog ergibt sich $f^C_{2,2,3} = X^2 + X + 1$. Ferner gilt $s := \frac{p^r-1}{p^t-1} = 5$. Dann ist, wie man durch Nachrechnen bestätigen kann, das Element $\alpha := \overline{X^5} = \overline{2X^3 + 5X^2 + 3X + 4}$ aus $\mathbb{Z}_8[X]/(X^4 + 4X^3 + 6X^2 + 3X + 1)$ eine Wurzel von $X^2 + X + 1$ und somit α ein Teichmüllererzeuger des zu $\mathrm{GR}(2^{2\cdot3}, 2^3)$ isomorphen Ringes $\mathbb{Z}_8[\alpha]$.

Wir schließen das Kapitel mit den folgenden Betrachtungen, die für eine einheitliche Darstellung aller endlichen Kettenringe im Computer äußerst nützlich sind:

Definition 1.14. Ist R ein Kettenring und $\sigma \in \mathrm{Aut}(R)$, so bezeichnet $R[X, \sigma]$ im Folgenden den Ring, dessen additive Gruppe derjenigen von $R[X]$ gleicht und dessen Multiplikation sich durch Fortsetzung von $X \cdot r := \sigma(r) \cdot X$ ergibt. Wir nennen $R[X, \sigma]$ den *Schiefpolynomring über R zum Automorphismus σ*.

Definition 1.15. Sei R ein Kettenring. Ein Polynom $f = \sum_{i=0}^{k} a_i X^i \in R[X]$ heißt *Eisensteinpolynom*, wenn für seine Koeffizienten gilt:

(i) $a_k = 1$.

(ii) $a_i \in \mathrm{Rad}(R)$ für $0 \le i \le k - 1$.

(iii) a_0 ist ein Erzeuger von $\mathrm{Rad}(R)$.

Satz 1.15. *Sei $R := GR(p^{rn}, p^n)$, $\sigma \in \mathrm{Aut}(R)$, $f \in R[X]$ ein Eisensteinpolynom vom Grad k und $s \in \mathbb{N}^*$ mit $(n - 1)k + 1 \le s \le nk$. Dann ist $S := R[X, \sigma]/(f, X^s)$ ein Kettenring mit $\theta_S = \overline{X}$, $m_S = s$, $p_S = p$ und $r_S = r$.*

Wichtig ist, dass auch die Umkehrung von Satz 1.15 gilt: Jeder Kettenring läßt sich in der dort vorausgesetzten Form darstellen, wobei R als Galoisring gewählt werden kann. Für kommutative Kettenringe wurde dies in [CD73] nachgewiesen, die allgemeine Version findet sich in [Nec73]:

Satz 1.16. *Sei R ein Kettenring mit den Parametern $q = p^r$ und Charakteristik p^n. Dann gibt es natürliche Zahlen k und t mit $1 \le t \le k$, $\sigma \in \mathrm{Aut}(GR(p^{rn}, p^n))$ sowie ein Eisensteinpolynom f in $GR(p^{rn}, p^n)[X]$, so dass R isomorph zu $GR(p^{rn}, p^n)[X, \sigma]/(f, X^s)$ ist, mit $s = (n - 1)k + t$.*

Kapitel 2.

Lineare Codes über Kettenringen

In diesem Kapitel wollen wir uns mit linearen Codes über Kettenringen beschäftigen. Da jeder Körper ein Kettenring der Kettenlänge 1 ist, stellen die klassischen linearen Codes über Körpern diesbezüglich einen Spezialfall dar. Wir werden sehen, dass einige aus der klassischen Situation vertraute Eigenschaften im allgemeinen Fall nicht mehr beziehungsweise nur unter entsprechender Anpassung der Definitionen, etwa beim Gewicht, gegeben sind. Für eine Einführung in die wichtigsten Grundlagen der Theorie der linearen Codes über Körpern sei auf die gängige Literatur verwiesen, etwa [BBF+06] und [HP03].

Innerhalb dieses Abschnitts sei R stets ein beliebiger Kettenring mit den Parametern $q = p^r$ und Kettenlänge m. Ferner sei θ ein Erzeuger von $\mathrm{Rad}(R)$.

Definition 2.1. Sei S ein endlicher Ring und M ein S-Linksmodul. M heißt *frei*, wenn M isomorph zu einer (nicht notwendigerweise endlichen) direkten Summe von Kopien von S ist. Gilt $M \cong S^k$, so heißt k der *Rang* von M.

Bemerkung 2.1. Enthält der Ring S aus obiger Definition ein Einselement, so kann man für jeden freien S-Linksmodul M von endlichem Rang k ein linear unabhängiges Erzeugendensystem der Mächtigkeit k angeben[1], also eine Menge $\mathcal{G} = \{g_0, g_1, \ldots, g_{k-1}\} \subsetneq M$, so dass:

(i) $M = \{\sum_{i=0}^{k-1} s_i g_i : s_i \in S \; \forall i\}$.

(ii) $\left[\sum_{i=0}^{k-1} s_i g_i = 0\right] \Rightarrow \forall i : s_i = 0$.

Definition 2.2. Ein *R-linearer Code* der *Länge* n ist ein Untermodul von ${}_R(R^n)$. Ist C frei, so wird C entsprechend auch als freier Code über R bezeichnet. C ist in diesem Fall isomorph zu ${}_R(R^k)$ mit $k \in \mathbb{N}^*$ geeignet; k heißt dann die *Dimension* von C.

Bemerkung 2.2.

Ist R kein Körper, so gibt es stets nicht-freie Untermoduln von ${}_R(R^n)$, zum Beispiel $\underbrace{\mathrm{Rad}(R) \times \mathrm{Rad}(R) \times \cdots \times \mathrm{Rad}(R)}_{n \text{ mal}}$.

[1] Besitzt S hingegen kein Einselement, so gilt dies im Allgemeinen nicht, z. B. $S := M := 2\mathbb{Z}_4$. M ist frei als $2\mathbb{Z}_4$-Modul, enthält jedoch kein linear unabhängiges Erzeugendensystem.

Für den Fall, dass C nicht frei ist, benötigen wir einen etwas allgemeineren Basisbegriff als den des linear unabhängigen Erzeugendensystems. Wir folgen dabei der Konvention aus [HL00]:

Definition 2.3. Sei C ein linearer Code über R. Eine endliche Folge $\mathcal{B} = (b_0, b_1, \ldots, b_{k-1})$ von Codeworten aus $C \setminus \{0\}$ nennen wir eine *Basis* von C, wenn gilt:

(i) $C = \{\sum_{i=0}^{k-1} r_i b_i : r_i \in R\}$.

(ii) $\left[\sum_{i=0}^{k-1} r_i b_i = 0\right] \Rightarrow \forall i : r_i b_i = 0$, also $C = \bigoplus_{i=0}^{k-1} R b_i$.

Bemerkung 2.3. In [HL00] wird eine Menge von b_i mit $[\sum_{i=0}^{k-1} r_i b_i = 0] \Rightarrow \forall i : r_i b_i = 0$ als *unabhängig* bezeichnet, während die stärkere Eigenschaft $[\sum_{i=0}^{k-1} r_i b_i = 0] \Rightarrow \forall i : r_i = 0$ dort *linear unabhängig* heißt.

Die folgende beiden Sätze stellen klar, dass jeder R-lineare Code eine Basis im Sinne von Definition 2.3 besitzt. Sie wurden in ähnlicher Form bereits in [NS00] gezeigt. Da die Beweise jedoch gleichermaßen instruktiv wie elementar sind, führen wir sie an dieser Stelle an:

Satz 2.1. *Sei $C \neq \{0\}$ ein R-linearer Code der Länge n und $\mathcal{V}_k$ für $k = 1, 2, \ldots, m-1$ ein Vertretersystem von $R/Rad(R)^k$. Dann ist C bis auf Spaltenvertauschungen der Zeilenraum einer Matrix*

$$\Gamma = \left(\begin{array}{ccccc|c} I_{t_m}\theta^0 & A_{m,m-1}\theta^0 & A_{m,m-2}\theta^0 & \ldots & A_{m,1}\theta^0 & B_m\theta^0 \\ 0 & I_{t_{m-1}}\theta^1 & A_{m-1,m-2}\theta^1 & \ldots & A_{m-1,1}\theta^1 & B_{m-1}\theta^1 \\ \vdots & \ddots & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & & 0 & I_{t_1}\theta^{m-1} & B_1\theta^{m-1} \end{array}\right),$$

wobei $t_1, t_2, \ldots, t_m \in \mathbb{N}$, $A_{i,j} \in \mathcal{V}_{i-j}^{t_i \times t_j}$ und $B_i \in R^{t_i \times (n - \sum_{j=1}^{m} t_j)}$.

Beweis. Sei $\Gamma^{(0)}$ eine $l \times n$-Matrix, deren Zeilen ein Erzeugendensystem von C bilden. Wir geben eine verallgemeinerte Form des Gauß-Algorithmus an, die $\Gamma^{(0)}$ schrittweise auf die gewünschte Form bringt, ohne den Zeilenraum (abgesehen von Spaltenvertauschungen) zu verändern:
Für $k = 0, 1, \ldots$ tue:

(i) Sind die Zeilen $k, k+1, \ldots, l-1$ von $\Gamma^{(k)}$ allesamt Nullzeilen, so setze Γ auf die Matrix, die sich durch Streichen dieser Zeilen aus $\Gamma^{(k)}$ ergibt. Γ hat dann die angegebene Form und der Algorithmus ist zu Ende.

(ii) Sonst:

(1) Wähle aus den Zeilen $k, k+1, \ldots, l-1$ von $\Gamma^{(k)}$ einen Eintrag $a_{i,j}$ mit minimaler Höhe s.

(2) Bringe $a_{i,j}$ durch Zeilen- und Spaltenvertauschungen an die Position (k,k).

(3) Multipliziere die k-te Zeile von links mit einer geeigneten Einheit von R, so dass der Eintrag an Position (k,k) den Wert θ^s hat.

(4) Bringe die k-te Spalte in den Zeilen $k+1,\ldots,l-1$ durch Subtraktion geeigneter Vielfacher der k-ten Zeile auf Null; dies ist möglich, da die Höhe dieser Einträge mindestens s beträgt.

(5) Für $i \in \{0,1,\ldots,k-1\}$: Sei t die Höhe des Eintrags an der Position (i,i). Dann ist $t \leq s$ und der Eintrag an Position (i,k) darstellbar als $r_1\theta^t$ mit $r_1 \in R$. Wähle $r_2 \in R$, so dass $r_1 - r_2\theta^{s-t} \in \mathcal{V}_{s-t}$. Ziehe das r_2-fache (von links) der k-ten Zeile von der i-ten Zeile ab.

(6) Setze $\Gamma^{(k+1)}$ auf die so entstandene Matrix, inkrementiere k und wiederhole.

□

Satz 2.2. *Für jeden R-linearen Code $C \neq \{0\}$ gibt es eine Basis $\mathcal{B} = (b_0, b_1, \ldots, b_{k-1})$ im Sinne von Definition 2.3 und eindeutig bestimmte natürliche Zahlen $k > 0$, $\lambda_0 \geq \lambda_1 \geq \ldots \lambda_{k-1} \geq 1$, so dass gilt:*

$$C \cong \bigoplus_{i=0}^{k-1} (R/Rad(R)^{\lambda_i}).$$

Beweis. Sei Γ eine Matrix, die nach Anwendung des Algorithmus aus Satz 2.1 auf ein beliebiges Erzeugendensystem von C entstand; ohne Einschränkung kann angenommen werden, dass dabei keine Spaltenvertauschungen nötig waren. Die Zeilen von Γ wollen wir mit $b_0, b_1, \ldots, b_{k-1}$ bezeichnen und λ_i sei die jeweils größte Periode, die unter den Komponenten von b_i auftritt. Dann fallen die λ_i monoton mit wachsendem i. Weiterhin sieht man leicht ein, dass die b_i eine Basis von C nach Definition 2.3 bilden: $C = \sum_{i=0}^{k-1} Rb_i$ ist klar und die Direktheit der Summe folgt sofort aus der Dreiecksgestalt im vorderen Teil von Γ zusammen mit der Bemerkung, dass b_{ii}, die i-te Komponente von b_i, gerade Periode λ_i besitzt und somit die Implikation

$$[r \in R, rb_{ii} = 0] \Rightarrow rb_i = 0$$

gilt. Ferner ist $f : R \to Rb_i$, $r \mapsto rb_i$ ein Epimorphismus von R-Linksmoduln mit dem Kern $\ker(f) = \mathrm{Rad}(R)^{\lambda_i}$, daher gilt $Rb_i \cong R/\mathrm{Rad}(R)^{\lambda_i}$ und insgesamt wie behauptet

$$C = \bigoplus_{i=0}^{k-1} Rb_i \cong \bigoplus_{i=0}^{k-1} (R/\mathrm{Rad}(R)^{\lambda_i}).$$

Es bleibt die Eindeutigkeit der λ_i und von k zu zeigen. Setzen wir, für $0 \leq j \leq m$, $a_j := |\{c \in C : \theta^j c = 0\}|$, so gilt

$$a_j = q^{\sum_{i=0}^{k-1} \min\{j,\lambda_i\}}$$

und damit

$$\mu_j := \log_q(a_j) = \sum_{i=0}^{k-1} \min\{j, \lambda_i\}.$$

Sei $\nu_j := |\{i : \lambda_i = j\}|$. Dann können die ν_j rekursiv berechnet werden mittels

$$\nu_j = \mu_j - \mu_{j-1} - \sum_{l=j+1}^{m} \nu_l \quad (j = m \ldots 1).$$

Die ν_j und damit natürlich auch die λ_i und k sind also schon durch die nur von C selbst abhängigen a_j bestimmt; somit ist die Eindeutigkeit dieser Werte nachgewiesen.

□

Bemerkung 2.4. Es gilt sogar das folgende Resultat [HL00]: Jeder endliche R-Linksmodul M ist direkte Summe zyklischer R-Moduln. Die Partition $\lambda = (\lambda_0, \ldots, \lambda_{k-1}) \vdash \log_q |M|$ mit

$${}_RM \cong R/\mathrm{Rad}(R)^{\lambda_0} \oplus \cdots \oplus R/\mathrm{Rad}(R)^{\lambda_{k-1}}$$

ist eindeutig durch M bestimmt. Genauer ist λ konjugiert zu $\mu = (\mu_0, \ldots, \mu_{m-1})$ mit $\mu_i := \dim \theta^i M/\theta^{i+1}M$.

Definition 2.4. Sei C ein R-linearer Code.

(i) Die Zahlentupel $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ aus Satz 2.2 bzw. $(t_m, t_{m-1}, \ldots, t_1)$ mit $t_i = |\{j : \lambda_j = i\}|$ nennen wir den *Umriss*[2] bzw. den *Typ* von C.

(ii) Zu einem Vektor $u \in R^n$ heißt die kleinste natürliche Zahl i mit $\theta^i u = 0$ die *Periode* von u. Entsprechend heißt $m - i$ die *Höhe* von u, wir werden sie im Folgenden mit $\mathfrak{h}(u)$ bezeichnen.

(iii) Bilden die Zeilen einer Matrix Γ eine Basis von C und ist dabei die Periode der i-ten Zeile λ_i, so nennen wir Γ eine *Generatormatrix* von C. Hat Γ darüber hinaus noch die Gestalt aus Satz 2.1, so sprechen wir von einer *systematischen* Generatormatrix.

(iv) Jede Spalte einer Generatormatrix liegt in der Menge

$$\mathcal{V}_\lambda := \left\{ v = \begin{pmatrix} v_0 \\ \vdots \\ v_{k-1} \end{pmatrix} \in R^k : v_i \in \mathrm{Rad}(R)^{m-\lambda_i} \right\}.$$

Entsprechend definieren wir

$$\mathcal{U}_\lambda := \{(\overline{u}_0, \overline{u}_1, \ldots, \overline{u}_{k-1}) : \overline{u}_i \in R/\mathrm{Rad}(R)^{\lambda_i}\}.$$

Elemente aus $\mathcal{V}_\lambda$ werden wir im Folgenden auch als *Spaltenvektoren* und Elemente aus $\mathcal{U}_\lambda$ (bzw. eigentlich genauer deren Repräsentanten) als *Zeilen-* oder *Informationsvektoren* bezeichnen.

[2]Dies ist der Versuch einer Übersetzung des englischen Wortes *shape*.

(v) Für $\overline{u} = (\overline{u}_0, \overline{u}_1, \ldots, \overline{u}_{k-1}) \in \mathcal{U}_\lambda$ und $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_{k-1})^T \in \mathcal{V}_\lambda$ sei

$$\overline{u}\gamma := (u_0, u_1, \ldots, u_{k-1})\gamma = \sum_{i=0}^{k-1} u_i\gamma_i$$

und für eine Generatormatrix Γ

$$\overline{u}\Gamma := (u_0, u_1, \ldots, u_{k-1})\Gamma.$$

Hierbei hängen die Werte von $\overline{u}\gamma$ und $\overline{u}\Gamma$ offensichtlich nicht von der Wahl der Repräsentanten u_i ab, sind also wohldefiniert.

Bemerkung 2.5. Die Elemente des von einer Generatormatrix Γ erzeugten Codes C lassen sich so darstellen: $C = \{\overline{u}\Gamma : \overline{u} \in \mathcal{U}_\lambda\}$.

2.1. Gewichtsfunktionen

Definition 2.5. Eine Funktion $w : R \to \mathbb{R}_0^+$ heißt eine *Gewichtsfunktion* oder auch kurz ein *Gewicht auf* R, wenn sie die folgenden Eigenschaften besitzt (vgl. [CH97]):

(i) Für alle $r \in R$: $w(r) = 0 \Leftrightarrow r = 0$.

(ii) w ist konstant auf zueinander assoziierten Elementen.

(iii) Für $r_1, r_2 \in R$ beliebig gilt: $w(r_1 + r_2) \leq w(r_1) + w(r_2)$.

Bemerkung 2.6. Erfüllt w nur die Eigenschaften (i) und (ii) der vorangehenden Definition, so sprechen wir von einem *Fastgewicht*[3]. Ist das Durchschnittsgewicht aller vom Nullideal verschiedenen Ideale gleich, so heißt w ein *homogenes* Gewicht bzw. ein *homogenes* Fastgewicht.

Definition 2.6. Sei w ein Gewicht und $n \in \mathbb{N}^*$. Wir definieren die Funktion

$$\pi_j^{(n)} : R^n \to R, v = (v_0, v_1, \ldots, v_{n-1}) \mapsto v_j,$$

als die *Projektion* von R^n auf die j-te Komponente. Die natürliche Verallgemeinerung von w auf R^n,

$$w^{(n)} : R^n \to \mathbb{R}_0^+, v \mapsto \sum_{j=0}^{n-1} w(\pi_j^{(n)}(v)),$$

ist das von w induzierte Gewicht auf R^n und entsprechend

$$d^{(n)} : R^n \to \mathbb{R}_0^+, d^{(n)}(v, v') := w^{(n)}(v - v'),$$

[3] In [CH97] wird ein solches w als *near-weight* bezeichnet.

die von w auf R^n induzierte Metrik[4]. Ist $C \leq R^n$ ein R-linearer Code, so heißt

$$d(C) := \min\{d^{(n)}(c, c') : c, c' \in C, c \neq c'\}$$

der *Minimalabstand* von C. Aufgrund der Linearität von C fällt er zusammen mit dem *Minimalgewicht* von C, also $\min\{w^{(n)}(c) : c \in C \setminus \{0\}\}$. Da n im weiteren Verlauf entweder aus dem Zusammenhang klar sein wird oder bewusst nicht spezifiziert werden soll, werden wir den Index bei π_j, w und d jeweils weglassen.

Es wäre nun naheliegend, bei den Codes über Kettenringen wie bei denen über Körpern die vom Hamminggewicht w_{Ham} induzierte Metrik d_{Ham} zu verwenden. Allerdings zeigt die folgende einfache Überlegung, dass die R-linearen Codes hierbei in der Regel recht schlecht abschneiden:

Ist nämlich $C \leq R^n$ ein R-linearer Code und $c \in C \setminus \{0\}$ ein Codewort, so erhält man durch Multiplikation von c mit einer geeigneten Potenz θ^l ein neues Codewort $c' \in C \setminus \{0\}$, bei dem alle Komponenten in $\text{Rad}(R)^{m-1}$ liegen. Das Hamminggewicht von c' ist dann höchstens so groß wie das von c. Folglich hat man für das Minimalgewicht von C in den Codeworten nur die Komponenten zu berücksichtigen, die maximale Periode besitzen, alle anderen zählen quasi nicht.

Eine weiteres Indiz, dass die Hammingmetrik nicht die günstigste Wahl für die Abstandsmessung bei den Codes über Kettenringen ist, ist folgende Beobachtung: Für einen nichtredundanten[5], k-dimensionalen linearen Code $C \leq \mathbb{F}_q^n$ gilt für jede Komponente $j \in \{0, 1, \ldots, n-1\}$ die Formel

$$\sum_{c \in C} w_{\text{Ham}}(\pi_j(c)) = (q-1)q^{k-1}.$$

Dass dies bei einem allgemeinen Kettenring nicht mehr der Fall ist, macht man sich leicht an dem Beispiel $R := \mathbb{Z}_4$, $C := \left\langle \begin{pmatrix} \overline{1} & \overline{2} \end{pmatrix} \right\rangle$ klar. Hier gilt

$$\sum_{c \in C} w_{\text{Ham}}(\pi_0(c)) = 3 \neq 2 = \sum_{c \in C} w_{\text{Ham}}(\pi_1(c)).$$

Es stellt sich nun die Frage, ob ein Gewicht existiert, dessen Metrik diese Eigenschaft auch auf die Codes über Kettenringen überträgt.

In [CH97] wurde ein homogenes Fastgewicht für die Restklassenringe $\mathbb{Z}_s$, $s \in \mathbb{N}^*$, eingeführt[6] und in [GS00] auf beliebige endliche Ringe verallgemeinert. Bei Kettenringen besitzt diese Verallgemeinerung genau die gewünschten Merkmale:

Definition 2.7. Die Funktion $w_{\text{hom}} : R \to \mathbb{R}_0^+$,

$$w_{\text{hom}}(r) := \begin{cases} 0 & \text{für } r = 0 \\ q & \text{für } r \in R^\times \theta^{m-1} \\ q-1 & \text{sonst} \end{cases},$$

[4]Die Metrikeigenschaften von $d^{(n)}$ folgen sofort aus denen des Gewichts w.

[5]Ein linearer Code C heißt *nichtredundant*, wenn in einer Generatormatrix von C keine Nullspalten vorkommen.

[6]Hierbei handelt es sich, falls $6 \nmid s$, sogar um ein homogenes Gewicht.

heißt das *homogene Gewicht* auf R.

Bemerkung 2.7.

(i) Man überzeugt sich leicht, dass w_{hom} ein Gewicht im Sinne von Definition 2.5 darstellt. Das folgende Lemma zeigt außerdem, dass die Bezeichnung *homogenes Gewicht* im Einklang mit der in Bemerkung 2.6 eingeführten Terminologie steht.

(ii) Ist R ein endlicher Körper $\mathbb{F}_q$, so gilt $w_{\text{hom}} = q \cdot w_{\text{Ham}}$.

Lemma 2.3. *Für das homogene Gewicht w_{hom} gilt:*

(i) In jedem Ideal $I \neq \{0\}$ von R beträgt das Durchschnittsgewicht $\overline{w}_I$ der Elemente aus I gerade $q-1$.

(ii) w_{hom} ist die bis auf Skalierung einzige homogene Gewichtsfunktion auf R.

(iii) Sei $C \leq R^n$ ein nichtredundanter, R-linearer Code. Für $j \in \{0, 1, \ldots, n-1\}$ ist

$$\sum_{c \in C} w_{hom}(\pi_j(c)) = |C|\,(q-1).$$

Beweis.

(i) Sei $I = \text{Rad}(R)^i$ mit $i < m$ und $J = \text{Rad}(R)^{m-1}$. Dann gilt

$$\overline{w}_I = \frac{1}{|I|} \sum_{r \in I} w_{\text{hom}}(r) = \frac{1}{q^{m-i}} \left(\sum_{r \in I \setminus J} w_{\text{hom}}(r) + \sum_{r \in J} w_{\text{hom}}(r) \right) =$$
$$= \frac{1}{q^{m-i}} \left((q^{m-i} - q)(q-1) + (q-1)q \right) = q - 1.$$

(ii) Sei w eine weitere homogene Gewichtsfunktion und w_{m-1} der Wert von w auf $R^\times \theta^{m-1}$. Dann beträgt das Durchschnittsgewicht von w auf $\text{Rad}(R)^{m-1}$ und damit auf allen Idealen von R gerade $\overline{w} = \frac{q-1}{q} w_{m-1}$. Durch Induktion nach i (für $i = 2 \ldots m$) sieht man leicht ein, dass w auf $R^\times \theta^{m-i} = \text{Rad}(R)^{m-i} \setminus \text{Rad}(R)^{m-(i-1)}$ den Wert $\overline{w}$ annehmen muss. Also ist w gerade das $\frac{w_{m-1}}{q}$-fache von w_{hom}.

(iii) Sei $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ der Umriss von C und Γ eine Generatormatrix, deren j-te Spalte wir mit $\gamma_j \in \mathcal{V}_\lambda$ bezeichnen. Die Abbildung

$$\phi_j : \mathcal{U}_\lambda \to {}_R R, \overline{u} \mapsto \overline{u}\gamma_j$$

ist ein wohldefinierter Homomorphismus zwischen R-Linksmoduln. Das Bild von ϕ_j, welches mit $\pi_j(C)$ identisch ist, ist daher das Ideal $I = \text{Rad}(R)^{\mathfrak{h}(\gamma_j)}$. Außerdem wird jeder Wert im Bild gleich häufig angenommen, also gerade $\frac{|C|}{|I|}$ mal. Somit gilt:

$$\sum_{c \in C} w_{\text{hom}}(\pi_j(c)) = |C| \frac{1}{|I|} \sum_{r \in I} w_{\text{hom}}(r) \overset{(i)}{=} |C|\,(q-1).$$

□

Wir werden uns in dieser Arbeit auf gute Codes bezüglich des homogenen Gewichts konzentrieren. Zu den bereits angeführten Gründen liefert die Existenz der im nächsten Kapitel behandelten Grayabbildung einen weiteren, denn sie erlaubt es, jeden R-linearen Code bezüglich des homogenen Gewichts isometrisch auf einen (im Allgemeinen nichtlinearen) Code über $\mathbb{F}_q$ der q^{m-1}-fachen Länge abzubilden.

2.2. Die Grayabbildung

In den Fokus der Forschung gerieten die linearen Codes über Kettenringen vor allem, nachdem Anfang der 90er Jahre in Arbeiten von Nechaev [Nec91] und Hammons u. a. [HKC+94] nachgewiesen wurde, dass sich einige Klassen sehr guter nichtlinearer Codes, wie etwa die Kerdock- und Preparata-Codes, als Bilder linearer Codes über $\mathbb{Z}_4$ unter Verwendung der zugehörigen Grayabbildung

$$\gamma : \mathbb{Z}_4 \to \mathbb{F}_2^2, \begin{cases} \overline{0} & \mapsto 00 \\ \overline{1} & \mapsto 01 \\ \overline{2} & \mapsto 11 \\ \overline{3} & \mapsto 10 \end{cases}$$

darstellen lassen. In [GS99] wurde die Grayabbildung so verallgemeinert:

Definition 2.8. Sei $\mathcal{A}$ eine Teichmüllermenge zu R, $u \in \mathbb{F}_q^q$ ein Vektor, der alle Elemente von $\mathbb{F}_q$ in einer beliebig festgelegten Reihenfolge enthält und $v = \mathbb{1}$ der Einsvektor in $\mathbb{F}_q^q$. Seien die m verschiedenen Vektoren w_0, w_1, …, w_{m-1} definiert durch:

$$\begin{aligned} w_0 &:= \underbrace{u \otimes v \otimes v \cdots \otimes v \otimes v}_{m-1 \text{ Faktoren}} \\ w_1 &:= v \otimes u \otimes v \cdots \otimes v \otimes v \\ &\vdots \\ w_{m-2} &:= v \otimes v \otimes v \cdots \otimes v \otimes u \\ w_{m-1} &:= v \otimes v \otimes v \cdots \otimes v \otimes v \end{aligned}$$

Jedes Element aus $(\mathbb{F}_q^q)^{\otimes m-1}$ wollen wir als Zeilenvektor in $\mathbb{F}_q^{q^{m-1}}$ interpretieren, indem wir den Koeffizienten zum Basisvektor $e_{i_{m-2}} \otimes e_{i_{m-3}} \otimes \cdots \otimes e_{i_0}$ an die Position mit der Nummer $\sum_{k=0}^{m-2} q^k i_k$ schreiben. Die Funktion

$$\bar{} : R \to R/\mathrm{Rad}(R) \cong \mathbb{F}_q, \bar{t} := t + \mathrm{Rad}(R)$$

sei die natürliche Projektion von R auf $\mathbb{F}_q$. Dann definieren wir mit Hilfe der θ-adischen Entwicklung eine Einbettung von R in $\mathbb{F}_q^{q^{m-1}}$ durch

$$\gamma : R \to \mathbb{F}_q^{q^{m-1}}, \sum_{i=0}^{m-1} \alpha_i \theta^i \mapsto \sum_{i=0}^{m-1} \overline{\alpha_i} w_i,$$

die sich durch komponentenweise Anwendung von γ problemlos auf die Einbettung $\gamma^{(n)}$ von R^n in $\mathbb{F}_q^{n(q^{m-1})}$ erweitern lässt. γ nennen wir die *Grayabbildung* zu R.

In [GS99] wird darauf verwiesen, dass es sich bei Bild(γ) um den Reed-Muller-Code erster Ordnung vom Grad $m-1$ über $\mathbb{F}_q$ handelt, dessen Gewichtszähler wohlbekannt ist. Die Gewichtsverteilung von Bild(γ) lässt sich jedoch auch ohne dieses Vorwissen auf elementarem Weg herleiten. Dazu verwenden wir als Hilfsmittel die folgende Definition und das anschließende Lemma:

Definition 2.9. Sei $s \in \mathbb{N}^*$. Einen Vektor $x = (x_0, x_1, \ldots, x_{q^s-1}) \in \mathbb{F}_q^{q^s}$ wollen wir für $0 \leq t < s$ als q^t*-alternierend* bezeichnen, wenn gilt:

(i) $\forall a \mid 0 \leq a \leq q^{s-t} - 1 : x_{aq^t} = x_{aq^t+1} = \cdots = x_{aq^t+(q^t-1)}$.

(ii) $\forall a \mid 0 \leq a \leq q^{s-(t+1)} - 1 : \{x_{aq^{t+1}}, x_{aq^{t+1}+q^t}, \ldots, x_{aq^{t+1}+(q-1)q^t}\} = \mathbb{F}_q$.

Bemerkung 2.8. Der Leser möge sich durch die kompliziert erscheinenden Indizes nicht abschrecken lassen. In Worten formuliert ist ein Vektor gerade dann q^t-alternierend, wenn seine Einträge von Beginn ab immer für jeweils q^t aufeinanderfolgende Stellen konstant bleiben und nach je q solchen Blöcken ganz $\mathbb{F}_q$ durchlaufen ist. Für $i \leq m-2$ sind die Vektoren w_i aus Definition 2.8 daher offensichtlich q^{m-2-i}-alternierend.

Lemma 2.4. *Seien $x, y \in \mathbb{F}_q^{q^s}$. Ist x q^{t_x}-alternierend und y q^{t_y}-alternierend mit $t_x < t_y$, so ist der Vektor $x + y$ q^{t_x}-alternierend.*

Beweis. Sei $z := x + y$. Wir rechnen die beiden Eigenschaften aus Definition 2.9 direkt nach:

(i) Sei $0 \leq a \leq q^{s-t_x} - 1$ und $a =: bq^{t_y-t_x} + c$ mit $0 \leq c < q^{t_y-t_x}$. Für $0 \leq i \leq q^{t_x} - 1$ gilt:

$$\begin{aligned} z_{aq^{t_x}+i} &= x_{aq^{t_x}+i} + y_{aq^{t_x}+i} \overset{x\ q^{t_x}\text{-alt.}}{=} x_{aq^{t_x}} + y_{bq^{t_y}+cq^{t_x}+i} \overset{y\ q^{t_y}\text{-alt.}}{=} x_{aq^{t_x}} + y_{bq^{t_y}+cq^{t_x}} = \\ &= x_{aq^{t_x}} + y_{aq^{t_x}} = z_{aq^{t_x}}. \end{aligned}$$

(ii) Sei $0 \leq a \leq q^{s-(t_x+1)} - 1$ und $a =: bq^{t_y-(t_x+1)} + c$ mit $0 \leq c < q^{t_y-(t_x+1)}$. Dann gilt, für $0 \leq i < q$:

$$\begin{aligned} z_{aq^{t_x+1}+iq^{t_x}} &= x_{aq^{t_x+1}+iq^{t_x}} + y_{aq^{t_x+1}+iq^{t_x}} = x_{aq^{t_x+1}+iq^{t_x}} + y_{bq^{t_y}+cq^{t_x+1}+iq^{t_x}} \overset{cq^{t_x+1}+iq^{t_x}<q^{t_y}}{=} \\ &= x_{aq^{t_x+1}+iq^{t_x}} + y_{bq^{t_y}} \overset{d:=y_{bq^{t_y}}}{=} x_{aq^{t_x+1}+iq^{t_x}} + d. \end{aligned}$$

Somit überträgt sich Eigenschaft (ii) von x auf z.

□

Lemma 2.5. *$C := Bild(\gamma)$ ist ein m-dimensionaler linearer Code der Länge q^{m-1} über $\mathbb{F}_q$ und besitzt den Gewichtszähler*

$$w_C(x) = (q-1)x^{q^{m-1}} + q(q^{m-1}-1)x^{(q-1)q^{m-2}} + x^0.$$

Beweis. Sei $c \in C$ mit $0 \neq c = \sum_{i=0}^{m-1} a_i w_i$. Ist $c' := c - a_{m-1}w_{m-1} \neq 0$, so ist die Zahl $\mu := \max\{i < m-1 : a_i \neq 0\}$ wohldefiniert, und wiederholte Anwendung von Lemma 2.4 ergibt, dass c' $q^{m-2-\mu}$-alternierend ist. Da Addition eines Vielfachen des Einsvektors w_{m-1} nichts ändert, ist auch c $q^{m-2-\mu}$-alternierend. Insbesondere kommt jedes Element aus $\mathbb{F}_q$ in c gleich häufig, nämlich gerade q^{m-2} mal, vor. c hat demnach das Gewicht $(q-1)q^{m-2}$. Ist dagegen $c' = 0$, so besitzt c als Vielfaches von w_{m-1} das Gewicht q^{m-1}. Damit ist die Formel für den Gewichtszähler bewiesen; aus ihr ergibt sich auch unmittelbar die lineare Unabhängigkeit der w_i und damit die Aussage zur Dimension von C.

□

Mit dem Wissen um den Gewichtszähler von Bild(γ) ist der folgende zentrale Satz aus [GS99] leicht zu beweisen:

Satz 2.6. *γ ist eine Isometrie zwischen $(R, q^{m-2} \cdot w_{hom})$ und $(\mathbb{F}_q^{q^{m-1}}, w_{Ham})$.*

Beweis. Seien $r, s \in R$ und $r = \sum_{i=0}^{m-1} \alpha_i \theta^i$, $s = \sum_{i=0}^{m-1} \beta_i \theta^i$ ihre θ-adischen Entwicklungen. Offensichtlich ist

$$d_{\text{hom}}(r,s) = 0 \Leftrightarrow r = s \Leftrightarrow d_{\text{Ham}}(\gamma(r), \gamma(s)) = 0.$$

Weiterhin gilt, da die $(q-1)$ Worte von Gewicht q^{m-1} in Bild(γ) Vielfache des Einsvektors sind:

$$\begin{aligned} d_{\text{hom}}(r,s) = q \Leftrightarrow [\alpha_i = \beta_i \text{ für } 0 \leq i < m-1 \ \wedge \alpha_{m-1} \neq \beta_{m-1}] \Leftrightarrow \\ \Leftrightarrow \gamma(r) - \gamma(s) \in {\mathbb{F}_q}^{\times} w_{m-1} \Leftrightarrow d_{\text{Ham}}(\gamma(r), \gamma(s)) = q^{m-1}. \end{aligned}$$

Die Äquivalenz

$$d_{\text{hom}}(r,s) = (q-1) \Leftrightarrow d_{\text{Ham}}(\gamma(r), \gamma(s)) = (q-1)q^{m-2}$$

ergibt sich dann automatisch.

□

Lemma 2.7.

(i) Für $\alpha \in \mathcal{A}$ und $r \in R$ gilt: $\gamma(\alpha r) = \overline{\alpha}\gamma(r)$.

(ii) γ ist genau dann additiv, wenn $\mathcal{A}$ additiv abgeschlossen ist und dies ist nach Satz 1.7 äquivalent zu $char(R) = p$.

(iii) Ist $C \leq R^n$ ein R-linearer Code und $\mathcal{A}$ additiv abgeschlossen, so ist $\gamma^{(n)}(C)$ ein linearer Code in $\mathbb{F}_q^{q^{m-1}}$.

Beweis.

(i) Ergibt sich durch einfaches Nachrechnen.

(ii) „$\Leftarrow$“: Sei $\mathcal{A}$ additiv abgeschlossen und seien $r, s \in R$ mit den θ-adischen Entwicklungen $r = \sum_{i=0}^{m-1} \alpha_i \theta^i$ und $s = \sum_{i=0}^{m-1} \beta_i \theta^i$. Dann ist $r + s = \sum_{i=0}^{m-1} (\alpha_i + \beta_i)\theta^i$ und dies ist wegen der Abgeschlossenheit von $\mathcal{A}$ auch gleichzeitig die θ-adische Entwicklung. Daher gilt:

$$\gamma(r+s) = \sum_{i=0}^{m-1} \overline{(\alpha_i + \beta_i)} w_i = \sum_{i=0}^{m-1} \overline{\alpha_i} w_i + \sum_{i=0}^{m-1} \overline{\beta_i} w_i = \gamma(r) + \gamma(s).$$

„$\Rightarrow$“: Falls $\mathcal{A}$ nicht additiv abgeschlossen ist, so gibt es $\alpha_1, \alpha_2 \in \mathcal{A}$ mit $\alpha_1 + \alpha_2 \notin \mathcal{A}$, etwa $\alpha_1 + \alpha_2 = \beta_0 + \underbrace{\sum_{i=1}^{m-1} \beta_i \theta^i}_{\neq 0}$ mit $\beta_i \in \mathcal{A}$. Da $\overline{\alpha_1 + \alpha_2} = \overline{\alpha_1} + \overline{\alpha_2} = \overline{\beta_0}$ folgt

$$\gamma(\alpha_1) + \gamma(\alpha_2) = (\overline{\alpha_1} + \overline{\alpha_2}) w_0 = \overline{\beta_0} w_0 \neq \sum_{i=0}^{m-1} \overline{\beta_i} w_i = \gamma(\alpha_1 + \alpha_2),$$

also die Behauptung.

(iii) Folgt direkt aus (i) und (ii).

□

Falls $\mathcal{A}$ nicht additiv abgeschlossen ist, so ist das Graybild eines R-linearen Codes nicht unbedingt ein linearer Code über $\mathbb{F}_q$, wie das folgende Beispiel zeigt:

Beispiel 2.1. Sei $R := \mathbb{Z}_8$ und $\theta := 2$. Es ist $p = q = 2$ und $m = 3$. Die Grayabbildung für diesen Fall lautet, mit $u := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$:

$$\gamma : \mathbb{Z}_8 \to \mathbb{F}_2^4, \begin{cases} \overline{0} & \mapsto 0000 \\ \overline{1} & \mapsto 0011 \\ \overline{2} & \mapsto 0101 \\ \overline{3} & \mapsto 0110 \\ \overline{4} & \mapsto 1111 \\ \overline{5} & \mapsto 1100 \\ \overline{6} & \mapsto 1010 \\ \overline{7} & \mapsto 1001 \end{cases}$$

Für den vom Vektor $\begin{pmatrix} \overline{1} & \overline{3} \end{pmatrix}$ erzeugten $\mathbb{Z}_8$-linearen Code C ist das Graybild die Menge

$$\{00000000, 00110110, 01011010, 01100011, 11111111, 11001001, 10100101, 10011100\}.$$

Diese ist nicht additiv abgeschlossen in $\mathbb{F}_2^8$, denn sie enthält beispielsweise nicht die Summe $00110110 + 01011010 = 01101100$.

2.3. Codes und Diophantische Ungleichungssysteme

In diesem Abschnitt soll ausgeführt werden, wie sich die Suche nach R-linearen Codes mit vorgegebenen Parametern in ein Diophantisches Ungleichungssystem übersetzen lässt, welches dann mit Hilfe der im nächsten Kapitel vorgestellten Heuristiken gelöst werden soll. w sei hier eine beliebige Gewichtsfunktion auf R und $w^{(n)}$ die entsprechende Erweiterung auf R^n. Das Ziel soll sein, zu $d \in \mathbb{N}^*$, vorgegebener Länge n und festem Umriss $(\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ einen entsprechenden R-linearen Code C zu finden, dessen Minimaldistanz mindestens d beträgt. Dazu werden wir versuchen, eine Generatormatrix $\Gamma = \left(\gamma_0 \mid \gamma_1 \mid \ldots \mid \gamma_{n-1} \right)$ von C zu konstruieren. Die Anforderung an das Minimalgewicht von C wird durch das Ungleichungssystem

$$\forall \overline{u} \in {\mathcal{U}_\lambda}^* := \mathcal{U}_\lambda \setminus \{0\} : w^{(n)}(\overline{u}\Gamma) = \sum_{j=0}^{n-1} w(\overline{u}\gamma_j) \geq d \tag{2.1}$$

zum Ausdruck gebracht. Die γ_j kann man hier als Variablen auffassen, die unabhängig voneinander mit Werten aus ${\mathcal{V}_\lambda}^* := \mathcal{V}_\lambda \setminus \{0\}$ belegt werden können[7]. Dieses Ungleichungssystem ist für $R \neq \mathbb{F}_2$ redundant, denn da w konstant auf zueinander assoziierten Elementen ist, genügt es, sich bei den Ungleichungen auf diejenigen zu einer Transversale der Bahnen der natürlichen Gruppenoperation von $R^\times$ auf ${\mathcal{U}_\lambda}^*$ per Linksmultiplikation zu beschränken. Analog lassen sich die Spalten von Γ durch beliebige andere Repräsentanten ihrer Bahn unter der Operation von $R^\times$ auf ${\mathcal{V}_\lambda}^*$ per Rechtsmultiplikation ersetzen, ohne dabei einen der Summanden in den Ungleichungen zu verändern. Auch die Reihenfolge der Spalten von Γ spielt keine Rolle. Wichtig ist lediglich, wie oft jeder einzelne Spaltenvektor vorkommt. Daher treffen wir folgende Definition:

Definition 2.10. Ab jetzt sei $\overline{\mathcal{U}_\lambda}$ stets eine Transversale von $R^\times \backslash\backslash {\mathcal{U}_\lambda}^*$ und $\overline{\mathcal{V}_\lambda}$ eine Transversale von $R^\times \backslash\backslash {\mathcal{V}_\lambda}^*$. Ferner sei $\mathfrak{u} := \left|\overline{\mathcal{U}_\lambda}\right|$ und $\mathfrak{v} := \left|\overline{\mathcal{V}_\lambda}\right|$. Jeden Vektor $x \in \mathbb{N}^{\mathfrak{v}}$, $x = (x_v)_{v \in \overline{\mathcal{V}_\lambda}}$, können wir identifizieren mit einer Generatormatrix Γ_x, wobei die Komponente x_v die Vielfachheit angibt, mit der v als Spalte in Γ_x auftritt. Umgekehrt lässt sich einer Generatormatrix Γ ein solcher Vektor x_Γ zuordnen, indem man x_v auf die Zahl von Spalten in Γ setzt, die unter Rechtsmultiplikation mit Einheiten aus R in derselben Bahn wie v liegen. Dies alles setzt natürlich voraus, dass eine Durchlaufreihenfolge für die Elemente von $\overline{\mathcal{V}_\lambda}$ festgelegt wurde, was wir stets als gegeben annehmen wollen.

Mit den vorangegangenen Überlegungen läßt sich 2.1 reduzieren auf das Diophantische Ungleichungssystem

$$\begin{aligned} \forall \overline{u} \in \overline{\mathcal{U}_\lambda} : \sum_{v \in \overline{\mathcal{V}_\lambda}} x_v w(\overline{u}v) &\geq d \\ \sum_{v \in \overline{\mathcal{V}_\lambda}} x_v &= n \end{aligned} \tag{2.2}$$

[7] Nullspalten wollen wir von vornherein ausschließen.

Die letzte Gleichung stellt sicher, dass eine Lösung immer aus genau n Spalten besteht. Stellen wir uns unter $\mathcal{M}^w = (m_{\overline{u}v})$ die $\mathfrak{u} \times \mathfrak{v}$-Matrix vor, deren Zeilen und Spalten in beliebiger, aber fester Reihenfolge mit den Elementen aus $\overline{\mathcal{U}_\lambda}$ beziehungsweise $\overline{\mathcal{V}_\lambda}$ indiziert sind, wobei $m_{\overline{u}v} = w(\overline{u}v)$, dann läßt sich das System mit $x \in \mathbb{N}^{\mathfrak{v}}$ schließlich kompakt schreiben als

$$\begin{aligned} \mathcal{M}^w x &\geq d \cdot \mathbb{1} \\ \mathbb{1}^T x &= n \end{aligned} \tag{2.3}$$

In Kapitel 3 werden wir uns mit Methoden zum Lösen dieses Systems beschäftigen.

Bemerkung 2.9. Aus den Erläuterungen in Definition 2.10 geht hervor, dass das Anfügen der Spalte $v \in \overline{\mathcal{V}_\lambda}$ an die Matrix Γ dem Inkrementieren von $x \in \mathbb{N}^{\mathfrak{v}}$ an der Komponente x_v, also der Setzung $x \leftarrow x + e_v$ entspricht, wobei e_v der Einheitsvektor mit Eins an Position v ist. Wir werden die beiden Sichtweisen im Weiteren, je nach Kontext, abwechselnd verwenden und hoffen, den Leser damit nicht zu verwirren. Außerdem werden wir manchmal etwas ungenau von den $v \in \overline{\mathcal{V}_\lambda}$ (statt von den eigentlich gemeinten x_v) als Variablen reden.

Jetzt wollen wir darauf eingehen, wie $\overline{\mathcal{U}_\lambda}$ und $\overline{\mathcal{V}_\lambda}$ konstruiert werden können:

Definition 2.11. Zu $v = (v_0, v_1, \ldots, v_{k-1})^T \in \mathcal{V}_\lambda$ sei

$$\iota(v) := \min\{i : \mathfrak{h}(v_i) = \mathfrak{h}(v)\}$$

die erste Position, an der ein Element minimaler Höhe im Vektor v auftritt.

Lemma 2.8.

(i) *$\mathfrak{h}$ und ι sind konstant unter der Operation von $R^\times$ auf ${\mathcal{V}_\lambda}^*$ per Rechtsmultiplikation.*

(ii) *In jeder Bahn $R^\times(v)$ unter dieser Gruppenoperation gibt es genau einen Vertreter v^* mit $v^*_{\iota(v)} = \theta^{\mathfrak{h}(v)}$.*

(iii) *Als Repräsentanten in $\overline{\mathcal{V}_\lambda}$ können somit die Vektoren gewählt werden, bei denen die erste Komponente mit minimaler Höhe eine Potenz von θ ist.*

Beweis.

(i) Da die Multiplikation mit einer Einheit die Höhe der Elemente von R unverändert lässt, ist die Aussage klar.

(ii) Sei $s = v_{\iota(v)}$. Da Elemente gleicher Höhe in R assoziiert sind, gibt es mindestens ein $r \in R^\times$ mit $sr = \theta^{\mathfrak{h}(v)}$. Sei nun $r' \in R^\times$ ein weiteres Element mit dieser Eigenschaft. Die anderen Komponenten von v haben mindestens die Höhe $\mathfrak{h}(v)$, lassen sich also darstellen in der Form $u\theta^t s$ mit $t \geq 0$ und $u \in R^\times$. Aus $(u\theta^t s)r = u\theta^{t+\mathfrak{h}(v)} = (u\theta^t s)r'$ ergibt sich dann $vr = vr'$ und damit die Behauptung.

(iii) Folgt direkt aus den ersten beiden Punkten.

□

Die Konstruktion von $\overline{\mathcal{U}_\lambda}$ lässt sich auf $\overline{\mathcal{V}_\lambda}$ zurückführen:

Lemma 2.9.

(i) Die Mengen $Rad(R)^{m-\lambda_i}$ und $R/Rad(R)^{\lambda_i}$ sind als R-Linksmoduln isomorph vermöge

$$\varphi_\lambda^{(i)} : Rad(R)^{m-\lambda_i} \to R/Rad(R)^{\lambda_i}, r\theta^{m-\lambda_i} \mapsto r + Rad(R)^{\lambda_i}.$$

(ii) $\mathcal{V}_\lambda$ und $\mathcal{U}_\lambda$ sind als R-Linksmoduln isomorph via

$$\varphi_\lambda : \mathcal{V}_\lambda \to \mathcal{U}_\lambda, (\dots, r_i\theta^{m-\lambda_i}, \dots)^T \mapsto (\dots, \varphi_\lambda^{(i)}(r_i\theta^{m-\lambda_i}), \dots).$$

Beweis. Es gilt:

$$\begin{aligned} r_1\theta^{m-\lambda_i} = r_2\theta^{m-\lambda_i} \Leftrightarrow (r_1 - r_2)\theta^{m-\lambda_i} = 0 \Leftrightarrow r_1 - r_2 \in \mathrm{Rad}(R)^{\lambda_i} \Leftrightarrow \\ \Leftrightarrow r_1 + \mathrm{Rad}(R)^{\lambda_i} = r_2 + \mathrm{Rad}(R)^{\lambda_i}. \end{aligned}$$

Das ergibt die Wohldefiniertheit und die Injektivität von $\varphi_\lambda^{(i)}$ in (i), die Surjektivität und die Homomorphieeigenschaft sind direkt aus der Definition klar. (ii) folgt sofort aus komponentenweiser Anwendung von Punkt (i).

□

Korollar 2.10. *Sei $\overline{\mathcal{V}_\lambda}$ die nach Lemma 2.8 konstruierte Transversale der Bahnen der Operation von $R^\times$ auf ${\mathcal{V}_\lambda}^*$ per Multiplikation von rechts. $\overline{\mathcal{V}_\lambda}$ ist, wie man durch eine völlig analoge Überlegung einsieht, auch eine Transversale der Operation von $R^\times$ auf ${\mathcal{V}_\lambda}^*$ per Linksmultiplikation.[8] Zusammen mit Punkt (ii) von Lemma 2.9 folgt daher, dass $\overline{\mathcal{U}_\lambda} := \varphi_\lambda(\overline{\mathcal{V}_\lambda})$ ein Repräsentantensystem von $R^\times \backslash\backslash {\mathcal{U}_\lambda}^*$ ist.*

Bemerkung 2.10. Im Folgenden werden wir, sofern nicht anders erwähnt, unter $\overline{\mathcal{V}_\lambda}$ beziehungsweise $\overline{\mathcal{U}_\lambda}$ immer die nach Lemma 2.8 und Korollar 2.10 konstruierten Repräsentantensysteme von $R^\times \backslash\backslash {\mathcal{V}_\lambda}^*$ beziehungsweise $R^\times \backslash\backslash {\mathcal{U}_\lambda}^*$ verstehen.

Beispiel 2.2. Sei $R = \mathbb{Z}_9$, $k = 2$, $\lambda_0 = 2$ und $\lambda_1 = 1$. Mit dem geschilderten Verfahren erhält man

$$\overline{\mathcal{V}_\lambda} = \left\{ \begin{pmatrix} \overline{1} \\ \overline{0} \end{pmatrix}, \begin{pmatrix} \overline{1} \\ \overline{3} \end{pmatrix}, \begin{pmatrix} \overline{1} \\ \overline{6} \end{pmatrix}, \begin{pmatrix} \overline{3} \\ \overline{0} \end{pmatrix}, \begin{pmatrix} \overline{3} \\ \overline{3} \end{pmatrix}, \begin{pmatrix} \overline{3} \\ \overline{6} \end{pmatrix}, \begin{pmatrix} \overline{0} \\ \overline{3} \end{pmatrix} \right\}$$

und

$$\overline{\mathcal{U}_\lambda} = \left\{ (\overline{1}, \overline{0}), (\overline{1}, \overline{1}), (\overline{1}, \overline{2}), (\overline{3}, \overline{0}), (\overline{3}, \overline{1}), (\overline{3}, \overline{2}), (\overline{0}, \overline{1}) \right\}.^9$$

[8] Für beliebige Bahntransversalen der beiden Operationen ist dies hingegen im Allgemeinen nicht richtig.

[9] Der einfacheren Lesbarkeit halber wurden für die Komponenten der Elemente aus $\overline{\mathcal{U}_\lambda}$ nur Repräsentanten notiert, der Vektor $(\overline{1}, \overline{0})$ steht also z. B. eigentlich für das Element $(\overline{1} + \{\overline{0}\}, \overline{0} + \{\overline{0}, \overline{3}, \overline{6}\})$.

$\overline{\mathcal{V}_\lambda}$ wollen wir für spätere Zwecke noch unterteilen:

Definition 2.12. Zu $0 \leq \vartheta \leq m$ und $0 \leq \zeta \leq k-1$ sei

$$\overline{\mathcal{V}_\lambda}(\vartheta, \zeta) := \{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(v) = \vartheta, \iota(v) = \zeta\}.$$

2.4. Modifikationen für die Suche nach Arcs

In diesem Abschnitt wollen wir zunächst die Interpretation von linearen Codes über endlichen Körpern $\mathbb{F}_q$ als Punktmenge in der projektiven Geometrie $\mathrm{PG}(k-1,q)$ erläutern. Wie wir sehen werden, besteht in diesem Fall eine Eins-zu-Eins-Beziehung zwischen den linearen Codes von Länge n und Minimaldistanz d und den sogenannten $(n, n-d)$-*Arcs*, das sind Punktmengen der Mächtigkeit n in $\mathrm{PG}(k-1,q)$, bei denen auf jeder Hyperebene höchstens $n-d$ Punkte liegen. Im Falle echter Kettenringe mit $m > 1$ dagegen besteht diese Beziehung nicht mehr, und die Suche nach Arcs unterscheidet sich grundsätzlich von der Suche nach Codes. Allerdings sind nur kleine Änderungen bei den Gleichungssystemen zu linearen Codes notwendig, um diese auf die Arcsuche anzupassen. Hierauf wird am Ende dieses Abschnitts eingegangen.

2.4.1. Die projektive Geometrie und Arcs über endlichen Körpern

Definition 2.13. Die Menge aller linearen Unterräume von $\mathbb{F}_q^k$ heißt die *projektive Geometrie der Dimension $k-1$ über $\mathbb{F}_q$* und wird mit $\mathrm{PG}(k-1,q)$ notiert. Die eindimensionalen Unterräume von $\mathbb{F}_q^k$ werden die *Punkte*, die $(k-1)$-dimensionalen die *Hyperebenen* in $\mathrm{PG}(k-1,q)$ genannt. Die Menge aller Punkte wollen wir mit $\mathcal{P}$ und die Menge aller Hyperebenen mit $\mathcal{H}$ bezeichnen. Gilt für einen Punkt p und eine Hyperebene h die Beziehung $p \subsetneq h$, so sagt man: „p liegt auf h“.

Sei $R = \mathbb{F}_q$ und entsprechend $\lambda = (1, 1, \ldots, 1) \vdash k$. Jeder Vektor $v \in \overline{\mathcal{V}_\lambda}$ korrespondiert genau mit einem Punkt $\mathbf{p}(v) := \langle v \rangle_{\mathbb{F}_q} \in \mathcal{P}$ als dessen Erzeuger. Auch für die Informationsvektoren aus $\overline{\mathcal{U}_\lambda}$ gibt es eine geometrische Deutung: Identifizieren wir $\overline{u} \in \overline{\mathcal{U}_\lambda}$ mit der Hyperebene $\mathbf{h}(\overline{u}) := u^\perp := \{v \in \mathbb{F}_q^k : uv = 0\}$, so gilt für alle $v \in \overline{\mathcal{V}_\lambda}$:

$$\overline{u}v = uv = 0 \Leftrightarrow \mathbf{p}(v) \text{ liegt auf } \mathbf{h}(\overline{u}).$$

Definition 2.14. Ein (n,u)-*Arc* in $\mathrm{PG}(k-1,q)$ ist eine Multimenge $\mathfrak{t}$ von Punkten aus $\mathcal{P}$ mit $|\mathfrak{t}| = n$, so dass auf jeder Hyperebene $h \in \mathcal{H}$ höchstens u Punkte[10] aus $\mathfrak{t}$ liegen.

Die bis auf Skalierung einzige mögliche Gewichtsfunktion auf $\mathbb{F}_q$ ist das Hamminggewicht

$$w_{\mathrm{Ham}} : \mathbb{F}_q \to \mathbb{R}_0^+, x \mapsto \begin{cases} 0 & \text{falls } x = 0, \\ 1 & \text{sonst.} \end{cases}$$

[10] gezählt mit Vielfachheiten

Aus den vorangegangenen Überlegungen ergibt sich bei Verwendung des Hamminggewichts direkt das folgende Lemma (vgl. zum Beispiel [HK99]):

Lemma 2.11. *Es sind äquivalent:*

(i) Es gibt eine Lösung des Systems (2.3) *zu* $R = \mathbb{F}_q$*, Dimension* k*,* $w = w_{Ham}$ *sowie rechter Seite* $d \cdot \mathbb{1}$ *bzw.* n*.*

(ii) Es gibt einen nichtredundanten linearen Code über $\mathbb{F}_q$ *mit Blocklänge* n*, Dimension* k *und minimaler Hammingdistanz* $\geq d$*.*

(iii) Es gibt einen $(n, n-d)$*-Arc in* $\mathrm{PG}(k-1, q)$*.*

Lösungen, Codes und Arcs in Lemma 2.11 gehen dabei wie folgt auseinander hervor: Jede Lösung zu (i) liefert die Generatormatrix Γ eines linearen Codes, der die Bedingungen in (ii) erfüllt, indem man den Vektor $v \in \overline{\mathcal{V}_\lambda}$ so oft als Spalte in Γ einträgt, wie die Variable x_v angibt. Umgekehrt liefert die Vielfachheit einer Spalte in der Generatormatrix eines Codes mit den Eigenschaften aus (ii) die notwendige Variablenbelegung für eine Lösung in (i). Der Übergang zwischen (ii) und (iii) ergibt sich, indem man zu jeder Spalte einer Generatormatrix in (ii) den davon erzeugten Punkt in den Arc bei (iii) aufnimmt und umgekehrt.

2.4.2. Verallgemeinerung auf endliche Kettenringe

Nun soll die Definition der projektiven Geometrie über endlichen Körpern auf allgemeine endliche Kettenringe übertragen werden. Dies leisten die folgenden Definitionen:

Definition 2.15. Sei R ein Kettenring und $k \geq 2$ eine natürliche Zahl. Die Menge aller *freien* Rechtsuntermoduln von R_R^k heißt die *projektive Hjelmslev-Geometrie* der Dimension $k-1$ über R und wird mit $\mathrm{PHG}(k-1, R)$ notiert. Die Menge dieser Untermoduln vom Rang eins heiße die *Punktmenge*, die der Untermoduln vom Rang $k-1$ die *Hyperebenenmenge* von $\mathrm{PHG}(k-1, R)$. Wie bei den Körpern erhält die Punktmenge das Symbol $\mathcal{P}$ und die Hyperebenenmenge das Symbol $\mathcal{H}$. Die Definition eines (n, u)-Arcs in $\mathrm{PHG}(k-1, R)$ ist völlig analog zu Definition 2.14.

Definition 2.16. Sei $k \in \mathbb{N}^*$. Ein Vektor $v \in R^k$ heißt *fett*, wenn v in mindestens einer Komponente eine Einheit aus R besitzt, also wenn $\mathfrak{h}(v) = 0$. Die Definition gilt gleichermaßen für Zeilen- wie Spaltenvektoren.

Sei nun $\lambda = (m, m, \ldots, m) \vdash mk$. Im Unterschied zur projektiven Geometrie über Körpern dürfen wir Punkte und Hyperebenen nun nicht mehr direkt mit den Vektoren aus $\overline{\mathcal{V}_\lambda}$ bzw. $\overline{\mathcal{U}_\lambda}$ gleichsetzen, da darin jeweils vom Nullvektor verschiedene, nichtfette Vektoren enthalten sind. Daher betrachten wir nun stattdessen die Mengen $\overline{\mathcal{V}_\lambda}^f := \overline{\mathcal{V}_\lambda} \cap \{v \in R^k : v \text{ fett}\}$ und $\overline{\mathcal{U}_\lambda}^f := \varphi_\lambda(\overline{\mathcal{V}_\lambda}^f)$. Wie oben können wir nun jeden Spaltenvektor $v \in \overline{\mathcal{V}_\lambda}^f$ mit dem Punkt $\mathbf{p}(v) := \langle v \rangle_R$ und jeden Zeilenvektor $\overline{u} \in \overline{\mathcal{U}_\lambda}^f$ mit der Hyperebene $\mathbf{h}(\overline{u}) := u^\perp := \{v \in R^k : uv = 0\}$ identifizieren und erhalten auf diese Weise auch alle Punkte bzw. Hyperebenen in PHG.

Die Analogie zu Lemma 2.11 für allgemeine Kettenringe lautet damit:

Lemma 2.12. *Sei $\lambda = (m, m, \ldots, m) \vdash mk$ und $\mathcal{M}_f^{w_{Ham}}$ die Matrix für das System 2.3 unter Verwendung des Hamminggewichts w_{Ham} und der Mengen $\overline{\mathcal{V}_\lambda{}^f}$ bzw. $\overline{\mathcal{U}_\lambda{}^f}$ (statt wie sonst $\overline{\mathcal{V}_\lambda}$ bzw. $\overline{\mathcal{U}_\lambda}$). Sei weiter $\overline{\mathfrak{v}} := \left|\overline{\mathcal{V}_\lambda{}^f}\right|$. Dann sind äquivalent:*

(i) Es gibt eine Lösung $x \in \mathbb{N}^{\overline{\mathfrak{v}}}$ des Systems $\mathcal{M}_f^{w_{Ham}} x \geq d \cdot \mathbb{1}$, $\mathbb{1}^T x = n$.

(ii) Es gibt einen $(n, n-d)$-Arc in $\mathrm{PHG}(k-1, R)$.

Die Korrespondenz zwischen Lösungen in Punkt (i) und Arcs in Punkt (ii) ist analog zu der bei Lemma 2.11.

2.5. Äquivalenz von R-linearen Codes

Wir wollen uns nun der Frage widmen, wann zwei verschiedene R-lineare Codes als äquivalent betrachtet werden können.

Definition 2.17. Sei C ein R-linearer Code der Länge n. Eine R-lineare Abbildung $\iota : C \to {}_R(R^n)$ heißt *lineare homogene Isometrie* auf C, wenn ι das homogene Gewicht erhält, also für alle $c \in C$ die Gleichung $w_{\mathrm{hom}}(\iota(c)) = w_{\mathrm{hom}}(c)$ erfüllt. Entsprechend nennen wir ι eine *lineare Hammingisometrie*, wenn sie das Hamminggewicht erhält.

Bemerkung 2.11. Für jede lineare homogene Isometrie beziehungsweise jede lineare Hammingisometrie ι gilt:

(i) ι ist injektiv.

(ii) ι ist abstandserhaltend bezüglich der vom homogenen Gewicht (beziehungsweise vom Hamminggewicht) induzierten Metrik, denn für $c_1, c_2 \in C$ gilt:

$$d_{\mathrm{hom}}(\iota(c_1), \iota(c_2)) = w_{\mathrm{hom}}(\iota(c_1) - \iota(c_2)) = w_{\mathrm{hom}}(\iota(c_1 - c_2)) = w_{\mathrm{hom}}(c_1 - c_2) = \\ = d_{\mathrm{hom}}(c_1, c_2) \text{ (analog für das Hamminggewicht).}$$

Definition 2.18. Eine Abbildung $\tau : {}_R(R^n) \to {}_R(R^n)$ heißt *monomiale Transformation*[11], wenn es $u = (u_0, u_1, \ldots, u_{n-1}) \in (R^\times)^n$ und $\pi \in S_n$ gibt mit

$$\tau((r_0, r_1, \ldots, r_{n-1})) = (r_{\pi^{-1}(0)} u_0, r_{\pi^{-1}(1)} u_1, \ldots, r_{\pi^{-1}(n-1)} u_{n-1}).$$

In der Notation identifizieren wir τ mit dem Paar (u, π).

Bemerkung 2.12. Jede monomiale Transformation ist offensichtlich sowohl linear homogene Isometrie als auch lineare Hammingisometrie.

[11] Genauer könnte man hier von einer *linksmonomialen* Transformation sprechen, um zu verdeutlichen, dass es sich um einen R-Linksmodulisomorphismus handelt.

Der folgende Satz wurde (für die allgemeineren Frobeniusringe) zunächst in [Woo99] mit Hilfe charaktertheoretischer Methoden gezeigt, während später in [GS00] ein kombinatorischer Beweis erschien.

Satz 2.13. *Sei C ein R-linearer Code der Länge n. Jede lineare Hammingisometrie $\iota : C \to {}_R(R^n)$ ist die Einschränkung einer monomialen Transformation von R^n.*

In [GS00] wurde (wieder für Frobeniusringe) auch noch der entsprechende Satz für das homogene Gewicht bewiesen:

Satz 2.14. *Sei C ein R-linearer Code der Länge n. Jede lineare homogene Isometrie $\iota : C \to {}_R(R^n)$ ist die Einschränkung einer monomialen Transformation von R^n.*

Bemerkung 2.13. Nach Satz 2.13 und 2.14 sind die linearen Hammingisometrien von einem Code C nach R^n gerade die linearen homogenen Isometrien und lassen sich stets auffassen als Einschränkungen monomialer Transformationen.

Da R-lineare Codes, die durch die genannten Isometrien ineinander überführt werden können, hinsichtlich ihrer codierungstheoretischen Eigenschaften (Gewicht und Abstand der Codeworte untereinander, insbesondere Minimaldistanz) völlig gleichwertig sind, ist es sinnvoll, diese Codes miteinander zu identifizieren. Das motiviert die folgende Definition.

Definition 2.19. Seien C_1 und C_2 zwei R-lineare Codes der Länge n. C_1 und C_2 heißen *linear äquivalent*, wenn eine monomiale Transformation $\tau = (u, \pi)$ auf R^n existiert mit $\tau(C_1) = C_2$.

Bemerkung 2.14. Definition 2.19 definiert eine Äquivalenzrelation auf den R-linearen Codes der Länge n. Da der Umriss eines Codes invariant unter monomialen Transformationen ist, können nur Codes von gleichem Umriss in einer Äquivalenzklasse liegen.

Satz und Definition 2.15. *Die Klassen linear äquivalenter Codes der Länge n sind gerade die Bahnen der Operation von $\mathcal{G}_n := R^\times \wr_n S_n$ auf den Untermoduln von R^n von links vermöge*

$$\mathcal{G}_n \ni ((u_0, \ldots, u_{n-1}), \pi) * (r_0, r_1, \ldots, r_{n-1}) := (r_{\pi^{-1}(0)} u_0^{-1}, \ldots, r_{\pi^{-1}(n-1)} u_{n-1}^{-1}).$$

Beweis. Es ist nur zu zeigen, dass es sich bei der oben definierten Abbildung

$$* : \mathcal{G}_n \times R^n \to R^n$$

tatsächlich um eine Gruppenoperation handelt:

(i) $1_{\mathcal{G}_n} * (r_0, \ldots, r_{n-1}) = ((1, 1, \ldots, 1), \mathrm{id}) * (r_0, \ldots, r_{n-1}) = (r_0, r_1, \ldots, r_{n-1})$ ist klar.

(ii) Seien $(u, \pi), (v, \rho) \in \mathcal{G}_n$ mit $u = (u_0, \ldots, u_{n-1})$ und $v = (v_0, \ldots, v_{n-1})$. Dann gilt, für $r = (r_0, \ldots, r_{n-1})$:

$$\begin{aligned}(u, \pi) * ((v, \rho) * r) &= (u, \pi) * (r_{\rho^{-1}(0)} v_0^{-1}, \ldots, r_{\rho^{-1}(n-1)} v_{n-1}^{-1}) = \\ &= (r_{\rho^{-1}(\pi^{-1}(0))} v_{\pi^{-1}(0)}^{-1} u_0^{-1}, \ldots, r_{\rho^{-1}(\pi^{-1}(n-1))} v_{\pi^{-1}(n-1)}^{-1} u_{n-1}^{-1}) = \\ &= (uv_\pi, \pi\rho) * (r_0, \ldots, r_{n-1}) = ((u, \pi)(v, \rho)) * (r_0, \ldots, r_{n-1}).\end{aligned}$$

□

Ein noch weiter reichender Äquivalenzbegriff ergibt sich bei Berücksichtigung sogenannter semilinearer Isometrien:

Definition 2.20. Seien M und N R-Linksmoduln. Eine Abbildung $\phi : M \to N$ heißt *semilinear*, wenn ein Ringautomorphismus $\sigma \in \mathrm{Aut}(R)$ existiert, so dass gilt:

(i) $\phi(m_1 + m_2) = \phi(m_1) + \phi(m_2)$ für alle $m_1, m_2 \in M$.

(ii) $\phi(rm) = \sigma(r)\phi(m)$ für alle $r \in R$, $m \in M$.

σ heißt dann ein *zu ϕ gehörender Ringautomorphismus*.[12]

Definition 2.21. Sei C ein R-linearer Code der Länge n. Eine semilineare Abbildung $\omega : C \to {}_R(R^n)$ heißt *semilineare homogene Isometrie* auf C, wenn ω das homogene Gewicht erhält; analog sind *semilineare Hammingisometrien* definiert.

Definition 2.22. Eine Abbildung $\tau : {}_R(R^n) \to {}_R(R^n)$ heißt *semimonomiale Transformation*[13], wenn es $u = (u_0, u_1, \ldots, u_{n-1}) \in (R^\times)^n$, $\pi \in S_n$ und $\sigma \in \mathrm{Aut}(R)$ gibt mit

$$\tau((r_0, r_1, \ldots, r_{n-1})) = (\sigma(r_{\pi^{-1}(0)})u_0, \sigma(r_{\pi^{-1}(1)})u_1, \ldots, \sigma(r_{\pi^{-1}(n-1)})u_{n-1}).$$

In der Notation identifizieren wir τ mit dem Tripel (u, π, σ).

Bemerkung 2.15. Jede semimonomiale Transformation $\tau = (u, \pi, \sigma)$ ist sowohl eine semilineare homogene Isometrie als auch eine semilineare Hammingisometrie, ein zu τ gehörender Ringautomorphismus ist dabei jeweils σ.

Satz 2.16. *Sei C ein R-linearer Code der Länge n. Jede semilineare Hammingisometrie (bzw. jede semilineare homogene Isometrie) $\omega : C \to {}_R(R^n)$ ist die Einschränkung einer semimonomialen Transformation von R^n. Insbesondere fallen also die Begriffe „semilineare Hammingisometrie“ und „semilineare homogene Isometrie“ zusammen.*

Beweis. Wir beweisen die Behauptung für Hammingisometrien, der Beweis für den homogenen Fall verläuft völlig identisch. Sei σ ein zu ω gehörender Ringautomorphismus. Wir führen die Behauptung auf den Fall der linearen Isometrien zurück. Wir verwenden dazu die semimonomialen Transformationen $\overline{\sigma} := (\mathbb{1}, \mathrm{id}, \sigma)$ und $\overline{\sigma^{-1}} := (\mathbb{1}, \mathrm{id}, \sigma^{-1})$, also

$$\overline{\sigma}(r_0, \ldots, r_{n-1}) = (\sigma(r_0), \ldots, \sigma(r_{n-1})) \text{ und } \overline{\sigma^{-1}}(r_0, \ldots, r_{n-1}) = (\sigma^{-1}(r_0), \ldots, \sigma^{-1}(r_{n-1})).$$

Die Abbildung $\delta := \overline{\sigma^{-1}} \circ \omega$ ist linear von C nach R^n: Die Additivität ist klar, außerdem gilt, für $s \in R$ und $c \in C$:

$$\delta(sc) = \overline{\sigma^{-1}}(\omega(sc)) = \sigma^{-1}(\sigma(s)\omega(c)) = s\sigma^{-1}(\omega(c)) = s\delta(c).$$

[12] Dabei ist σ nicht unbedingt eindeutig bestimmt, wie man sich leicht anhand des Beispiels $\phi \equiv 0$ klarmacht.

[13] Auch hier könnte man präziser wieder von einer linkssemimonomialen Transformation sprechen.

Weiterhin ist δ als Komposition von Hammingisometrien wieder eine Hammingisometrie. Nach Satz 2.13 existiert also eine monomiale Transformation $\Delta : R^n \to R^n$ mit $\Delta_{|C} = \delta$. Setzen wir nun $\Omega := \overline{\sigma} \circ \Delta$, so ist Ω offensichtlich eine semimonomiale Transformation. Außerdem gilt

$$\Omega_{|C} = \overline{\sigma} \circ \Delta_{|C} = \overline{\sigma} \circ \delta = \overline{\sigma} \circ (\overline{\sigma^{-1}} \circ \omega) = \omega,$$

und somit ist die Behauptung gezeigt.

□

Wie schon bei den linearen Isometrien definieren wir:

Definition 2.23. Seien C_1 und C_2 zwei R-lineare Codes der Länge n. C_1 und C_2 heißen *semilinear äquivalent*, wenn eine semimonomiale Transformation $\tau = (u, \pi, \sigma)$ auf R^n existiert mit $\tau(C_1) = C_2$.

Bemerkung 2.16. Auch durch Definition 2.23 wird eine Äquivalenzrelation auf den R-linearen Codes der Länge n definiert, die eine Vergröberung gegenüber der linearen Äquivalenz darstellt; der Umriss der Codes ist jedoch weiterhin konstant auf den Äquivalenzklassen.

Satz und Definition 2.17. *Die Klassen semilinear äquivalenter Codes der Länge n sind gerade die Bahnen der Operation von $\mathcal{H}_n := (R^\times)^n \rtimes (S_n \times \mathrm{Aut}(R))$ auf den Untermoduln von R^n von links vermöge*

$$\mathcal{H}_n \ni ((u_0, \dots, u_{n-1}), (\pi, \sigma)) * (r_0, r_1, \dots, r_{n-1}) := (\sigma(r_{\pi^{-1}(0)})u_0^{-1}, \dots, \sigma(r_{\pi^{-1}(n-1)})u_{n-1}^{-1}).$$

Beweis. Wieder sind nur die Eigenschaften einer Gruppenoperation nachzuweisen:

(i) $1_{\mathcal{H}_n} * (r_0, \dots, r_{n-1}) = ((1, 1, \dots, 1), (\mathrm{id}, \mathrm{id})) * (r_0, \dots, r_{n-1}) = (r_0, r_1, \dots, r_{n-1})$ ist klar.

(ii) Sei $u = (u_0, \dots, u_{n-1})$, $v = (v_0, \dots, v_{n-1})$ und $h_1 = (u, (\pi, \sigma)), h_2 = (v, (\rho, \tau)) \in \mathcal{H}_n$. Für $r = (r_0, \dots, r_{n-1})$ gilt:

$$\begin{aligned} h_1 * (h_2 * r) &= (u, (\pi, \sigma)) * (\dots, \tau(r_{\rho^{-1}(i)})v_i^{-1}, \dots) = \\ &= (\dots, \sigma(\tau(r_{\rho^{-1}(\pi^{-1}(i))})v_{\pi^{-1}(i)}^{-1})u_i^{-1}, \dots) = \\ &= (\dots, (\sigma \circ \tau)(r_{(\pi\circ\rho)^{-1}(i)})(u_i\sigma(v_{\pi^{-1}(i)}))^{-1}, \dots) = \\ &= (uv_{(\pi,\sigma)}, (\pi \circ \rho, \sigma \circ \tau)) * (\dots, r_i, \dots) = \\ &= ((u, (\pi, \sigma))(v, (\rho, \tau))) * (\dots, r_i, \dots). \end{aligned}$$

□

Da der später vorgestellte Algorithmus nicht mit den Codes selbst, sondern mit deren Generatormatrizen arbeitet, müssen wir den Äquivalenzbegriff entsprechend übertragen.

Definition 2.24. Zwei Generatormatrizen Γ_1 und Γ_2 heißen *linear (semilinear) äquivalent*, wenn die von ihnen erzeugten Codes linear (semilinear) äquivalent sind.

Unter der Anwendung einer (semi-)monomialen Transformation τ auf eine Generatormatrix Γ wollen wir die Anwendung von τ auf die einzelnen Zeilenvektoren von Γ verstehen. Natürlich sind Generatormatrizen, die auf diese Weise auseinander hervorgehen, (semi-)linear äquivalent. Umgekehrt können Generatormatrizen aber (semi-)linear äquivalent sein, ohne dass sie durch eine (semi-)monomiale Transformation ineinander überführbar sind. Zum Beispiel erzeugen die Matrizen

$$\Gamma_1 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ und } \Gamma_2 := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

über einem beliebigen endlichen Kettenring R beide ganz R^2 und sind daher nach Definition 2.24 linear äquivalent, können aber offensichtlich nicht durch eine semilineare Transformation aufeinander abgebildet werden. Dies zeigt, dass die Menge der semilinearen Transformationen als operierende Gruppe zu klein ist. Einer Lösung dieses Problems wollen wir uns mit den folgenden Überlegungen widmen:

Lemma 2.18. *Sei M eine $l \times l$-Matrix über R. Folgende Aussagen sind äquivalent:*

(i) M ist invertierbar.

(ii) Multiplikation von Zeilenvektoren von links an M erhält deren Höhe.

(iii) Multiplikation fetter Zeilenvektoren von links an M liefert wieder fette Vektoren.

Die Aussage gilt genauso, wenn man „rechts“ durch „links“ und „Zeilenvektor“ durch „Spaltenvektor“ ersetzt.

Beweis.

$(i) \Rightarrow (ii)$**:** Sei M invertierbar, M^{-1} ihre Inverse und $u \in R^l$ ein Zeilenvektor mit Höhe i. Sei j die Höhe des Vektors uM. Da Matrixmultiplikation die Höhe höchstens vergrößert, gilt $i \leq j$. Umgekehrt lässt sich u aber auch schreiben als $u = (uM)M^{-1}$ und mit demselben Argument folgt $i \geq j$, also insgesamt $i = j$.

$(ii) \Rightarrow (i)$**:** Die Multiplikation der Zeilenvektoren aus R^l mit M von rechts ist ein Endomorphismus von R^l. Aus der Voraussetzung folgt, dass dieser trivialen Kern besitzt. Aufgrund der Endlichkeit von R^l handelt es sich damit um einen Automorphismus. Das ist gleichbedeutend mit der Invertierbarkeit von M.

$(ii) \Rightarrow (iii)$**:** Ist klar.

$(iii) \Rightarrow (ii)$**:** Sei u ein Zeilenvektor der Höhe i. Dann läßt sich u schreiben als $\theta^i u'$, wobei u' ein fetter Vektor ist. Es folgt $uM = (\theta^i u')M = \theta^i(u'M)$. $u'M$ ist nach Voraussetzung fett und uM hat somit Höhe i.

Der Beweis verläuft völlig analog bei Ersetzen von „rechts“ durch „links“ beziehungsweise „Zeilenvektor“ durch „Spaltenvektor“.

□

Lemma 2.19. *Sei M eine invertierbare $l \times l$-Matrix über R und N eine weitere Matrix mit gleichen Dimensionen, deren Einträge alle aus $Rad(R)$ stammen. Dann ist auch $M + N$ invertierbar.*

Beweis. N lässt sich schreiben in der Form $N = N'\theta$ mit $N' \in R^{l\times l}$ geeignet. Sei $u \in R^l$ ein fetter Vektor. Dann handelt es sich, da uM gemäß Lemma 2.18 fett ist, bei $u(M+N) = u(M+N'\theta) = uM+uN'\theta$ ebenfalls um einen fetten Vektor[14]. Wieder nach Lemma 2.18 folgt damit die Invertierbarkeit von $M + N$.

□

Lemma 2.20. *Sei $r \in \mathbb{N}^*$ und $l_i \in \mathbb{N}^*$ für $0 \leq i \leq r-1$. Eine Matrix $A \in R^{l\times l}$ der Form*

$$A = \begin{pmatrix} A_{0,0} & * & \dots & * \\ A_{1,0} & A_{1,1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ A_{r-1,0} & \dots & A_{r-1,r-2} & A_{r-1,r-1} \end{pmatrix}$$

mit $A_{i,i} \in R^{l_i\times l_i}$ und $A_{i,j} \in Rad(R)^{l_i\times l_j}$ für $i > j$ ist genau dann invertierbar, wenn alle $A_{i,i}$ invertierbar sind.

Beweis. Nach Lemma 2.19 kann ohne Einschränkung angenommen werden, dass, für $i > j$, alle $A_{i,j}$ Nullmatrizen sind.

(i) „⇒“: Sei A invertierbar. Wie zeigen die Invertierbarkeit von $A_{i,i}$ durch Induktion nach i:
$\underline{i = 0:}$ Klar.
$\underline{i - 1 \to i:}$ Sei $u_i \in R^{l_i}$ mit $A_{i,i}u_i = 0$. Die Induktionsvoraussetzung liefert durch sukzessives Auflösen die Existenz eines Vektors $v = (v_0, \dots, v_{i-1}, v_i = u_i, 0, \dots, 0)$ aus R^l mit $Av = 0$. Wegen der Invertierbarkeit von A folgt $v = 0$ und insbesondere $u_i = 0$. $A_{i,i}$ ist damit invertierbar.

(ii) „⇐“: Seien alle $A_{i,i}$ invertierbar und $u = (u_0, u_1, \dots, u_{r-1}) \in R^l$ mit $u_i \in R^{l_i}$ und $uA = 0$. Induktion nach i liefert sofort, dass stets $u_iA_i = 0$ und damit auch $u_i = 0$, insgesamt also $u = 0$ gilt, was die Invertierbarkeit von A zeigt.

□

Definition 2.25. Sei $n \in \mathbb{N}^*$, $\lambda = (\lambda_0, \lambda_1, \dots, \lambda_{k-1})$ ein Umriss und $(t_m, t_{m-1}, \dots, t_1)$ der zu λ gehörige Typ. Dann bezeichne

(i) G_n^λ die Menge der Generatormatrizen aus $R^{k\times n}$ mit Umriss λ.

[14] vgl. Lemma 1.4

(ii) $\mathcal{A}^\lambda$ die Menge der quadratischen Matrizen A der Form

$$A = \begin{pmatrix} A_{0,0} & A_{0,1} & \dots & & & & A_{0,m-1} \\ A_{1,0}\theta^1 & A_{1,1} & A_{1,2} & \dots & & & A_{1,m-1} \\ \vdots & \ddots & \ddots & & & & \vdots \\ A_{i,0}\theta^i & \dots & A_{i,i-1}\theta^1 & A_{i,i} & A_{i,i+1} & \dots & A_{i,m-1} \\ \vdots & & & \ddots & \ddots & & \vdots \\ \vdots & & & & \ddots & \ddots & \vdots \\ A_{m-1,0}\theta^{m-1} & \dots & & & & A_{m-1,m-2}\theta^1 & A_{m-1,m-1} \end{pmatrix},$$

wobei jedes $A_{i,j}$ eine $t_{m-i} \times t_{m-j}$-Matrix über R ist und die $A_{i,i}$ zusätzlich invertierbar sind.

Lemma 2.21. *$\mathcal{A}^\lambda$ ist eine Untergruppe von $GL(k, R)$.*

Beweis. Jedes $A \in \mathcal{A}^\lambda$ ist wegen der Invertierbarkeit der $A_{i,i}$ nach Lemma 2.20 auch selbst invertierbar. Seien nun $A, B \in \mathcal{A}^\lambda$ beliebig und $C = AB$. $A_{i,j}, B_{i,j}$ und $C_{i,j}$ seien die entsprechenden Teilmatrizen. Für $i \geq j$ gilt dann:

$$C_{i,j} = \sum_{l=0}^{j} A_{i,l}B_{l,j}\theta^{i-l} + \sum_{l=j+1}^{i} A_{i,l}\theta^{i-l}B_{l,j}\theta^{l-j} + \sum_{l=i+1}^{m-1} A_{i,l}B_{l,j}\theta^{l-j}.$$

Die Einträge aller Summanden haben Höhe $\geq i - j$, also auch die von $C_{i,j}$. Für den Fall $i = j$ entfällt die mittlere Summe, und abgesehen von $A_{i,i}B_{i,i}$ treten nur Summanden mit Einträgen aus $\mathrm{Rad}(R)$ auf. Nach Voraussetzung ist $A_{i,i}B_{i,i}$ invertierbar und damit, wegen Lemma 2.19, auch $C_{i,i}$. C ist also wieder von derselben Gestalt wie A und B, $\mathcal{A}^\lambda$ daher abgeschlossen unter Multiplikation und als endliche Teilmenge der Gruppe $\mathrm{GL}(k, R)$ eine Untergruppe derselben.

□

Satz 2.22. *Sei C ein R-linearer Code und $\lambda = (\lambda_0, \lambda_1, \dots, \lambda_{k-1})$ sein Umriss. Weiter sei Γ eine Generatormatrix von C. Dann ist die Menge aller Generatormatrizen von C gerade die Bahn von Γ unter der Operation von $\mathcal{A}^\lambda$ auf G_n^λ per Multiplikation von links.*

Beweis.

(i) Sei $A \in \mathcal{A}^\lambda$ beliebig. Wegen

$$C = \{u\Gamma : u \in R^k\} \overset{A \text{ inv.}}{=} \{(uA)\Gamma : u \in R^k\} = \{u(A\Gamma) : u \in R^k\}$$

erzeugen auch die Zeilen von $A\Gamma$ den Code C. Sei π_i die Periode der i-ten Zeile von $A\Gamma$. Aufgrund der Struktur von A und Γ ergibt sich sofort: $\pi_i \leq \lambda_i$. Aus $|C| = q^{\sum_{i=0}^{k-1} \lambda_i} = q^{\sum_{i=0}^{k-1} \pi_i}$ folgt dann automatisch $\pi_i = \lambda_i$ und die Direktheit der Summe der von den einzelnen Zeilen von $A\Gamma$ erzeugten Untermoduln. $A\Gamma$ ist also wieder eine Generatormatrix von C.

(ii) Es bleibt zu zeigen, dass alle Generatormatrizen von C in der Bahn $\mathcal{A}^\lambda(\Gamma)$ liegen. Hierbei können wir ohne Beschränkung der Allgemeinheit annehmen, dass Γ eine systematische Generatormatrix nach Satz 2.1 ist (bis auf evtl. nötige Spaltenvertauschungen liegt eine solche in $\mathcal{A}^\lambda(\Gamma)$). Sei nun Γ' eine beliebige weitere Generatormatrix von C. Da die Zeilen von Γ und Γ' Codeworte aus C sind, gibt es Matrizen $A, B \in R^{k\times k}$ mit $A\Gamma = \Gamma'$ und $B\Gamma' = \Gamma$. Es folgt $BA\Gamma = \Gamma$, was äquivalent ist zu $(BA - I_k)\Gamma = 0$. Die Zeilen von $D := BA - I_k$ liegen also im Linkskern von Γ und haben deshalb, wie man sich leicht überlegt, Periode $\leq m-1$. Wegen $BA = I_k + D$ folgt mit Lemma 2.19, dass BA invertierbar ist und somit insbesondere auch A. Es bezeichne a_{ij} den Eintrag von A in der i-ten Zeile und j-ten Spalte. Da die i-te Zeile von Γ' Periode λ_i besitzt, gilt $a_{ij} \in \mathrm{Rad}(R)^{\lambda_j-\lambda_i}$ für $i > j$. Wegen Lemma 2.20 sind mit A auch die Diagonalblöcke invertierbar. Darum gilt $A \in \mathcal{A}^\lambda$ und es ist alles gezeigt.

□

Lemma 2.23. *Für jede Generatormatrix Γ zum Umriss $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ gilt für den Stabilisator $\mathcal{A}^\lambda_\Gamma$ unter der Operation von $\mathcal{A}^\lambda$:*

$$\mathcal{A}^\lambda_\Gamma = \mathcal{S}^\lambda := \{A \in R^{k\times k} : A = I_k + \left(\begin{array}{c|c|c|c} a_0\theta^{\lambda_0} & a_1\theta^{\lambda_1} & \ldots & a_{k-1}\theta^{\lambda_{k-1}} \end{array} \right), a_i \in R^k\}$$

Insbesondere ist $\mathcal{S}^\lambda$ (als Schnitt aller elementweisen Stabilisatoren) Normalteiler von $\mathcal{A}^\lambda$.

Beweis. Dass jede Matrix $A \in \mathcal{S}^\lambda$ im Stabilisator von Γ liegt, ist klar. Sei umgekehrt $T \in \mathcal{A}^\lambda$ eine Matrix in $\mathcal{A}^\lambda_\Gamma$ und T_{i*} deren i-te Zeile. Dann gilt: $(e_i - T_{i*})\Gamma = 0$. Da die Zeilen von Γ eine Basis bilden, folgt daraus

$$(e_i - T_{i*}) = (r_{i,0}\theta^{\lambda_0}, r_{i,1}\theta^{\lambda_1}, \ldots, r_{i,k-1}\theta^{\lambda_{k-1}})$$

mit $r_{i,j} \in R$ geeignet. Da dies für alle Zeilen von T gilt, folgt $T \in \mathcal{S}^\lambda$.

□

Korollar 2.24. *Sei $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ ein Umriss und seien $\mathcal{A}^\lambda$, $\mathcal{S}^\lambda$, $\mathcal{H}_n$ und G^λ_n wie zuvor definiert. Die semilinearen Äquivalenzklassen aller R-linearen Codes der Länge n vom Umriss λ lassen sich identifizieren mit den Bahnen*

$$\mathcal{H}_n \backslash\backslash ((\mathcal{A}^\lambda/\mathcal{S}^\lambda) \backslash\backslash G^\lambda_n),$$

wobei $\mathcal{H}_n$ auf $(\mathcal{A}^\lambda/\mathcal{S}^\lambda)\backslash\backslash G^\lambda_n$ vermöge

$$h * (\mathcal{A}^\lambda/\mathcal{S}^\lambda)(\Gamma) := (\mathcal{A}^\lambda/\mathcal{S}^\lambda)\left(\begin{array}{c} h * \Gamma_{0*} \\ \hline h * \Gamma_{1*} \\ \hline \vdots \\ \hline h * \Gamma_{k-1*} \end{array} \right)$$

operiert.

Beweis. Es ist nur die Wohldefiniertheit zu zeigen. Offensichtlich spielt die Wahl der Repräsentanten für die Elemente aus $\mathcal{A}^\lambda/\mathcal{S}^\lambda$ keine Rolle. Seien nun $\Gamma, \overline{\Gamma} \in G_n^\lambda$ mit $\overline{\Gamma} = A * \Gamma$ für $A \in \mathcal{A}^\lambda$ geeignet, γ_j die j-te Spalte von Γ und $h = (u, (\pi, \sigma)) \in \mathcal{H}_n$. Dann gilt für die j-te Spalte von $h * \overline{\Gamma}$:

$$(h * \overline{\Gamma})_j = \sigma(A\gamma_{\pi^{-1}(j)})u_j^{-1} = \sigma(A)\sigma(\gamma_{\pi^{-1}(j)})u_j^{-1} = \sigma(A)(h * \Gamma)_j.$$

Dies ist richtig für alle j, daher haben wir $h * \overline{\Gamma} = \sigma(A)(h * \Gamma)$. Man überzeugt sich leicht, dass mit A auch $\sigma(A)$ Element von $\mathcal{A}^\lambda$ ist. Folglich liegen $h * \Gamma$ und $h * \overline{\Gamma}$ in einer Bahn unter $\mathcal{A}^\lambda/\mathcal{S}^\lambda$ und die obige Setzung ist wohldefiniert.

□

Kapitel 3.

Heuristisches Lösungsverfahren

Auch in diesem Kapitel, welches zusammen mit dem nächsten den Kern der vorliegenden Arbeit bildet, sei R wieder ein beliebiger endlicher Kettenring mit den üblichen Bezeichnern für die Parameter und w eine fest gewählte Gewichtsfunktion auf R. Wir werden eine heuristische Methode zum Lösen des Diophantischen Ungleichungssystems

$$\mathcal{M}^w x \geq d \cdot \mathbb{1}$$
$$\mathbb{1}^T x = n$$

aus Abschnitt 2.3 entwickeln, welche eine Verallgemeinerung des in [Zwa07] beziehungsweise [Zwa08] beschriebenen Verfahrens darstellt. Sie basiert auf dem Prinzip, die Variablen, ausgehend von $x = \vec{0}$,[1] sukzessive so zu erhöhen, dass dabei in jedem Schritt eine Bewertungsfunktion

$$\mathcal{E} : \mathbb{N}^{\mathfrak{v}} \to \mathbb{R}_0^+, x \mapsto \mathcal{E}(x)$$

maximiert wird. Daher handelt es sich um einen sogenannten *Greedy-Algorithmus*. Derartige Verfahren wurden in der Vergangenheit schon häufig mit Erfolg eingesetzt. In [Es98] beispielsweise wurde mit einem Greedy-Algorithmus nach möglichst großen Unabhängigkeitsmengen in einem ungerichteten Graphen gesucht, um *constant-weight codes*[2] hoher Kardinalität zu konstruieren. Die individuelle Komponente dieser Ansätze liegt in der jeweiligen Definition von $\mathcal{E}$, hier sind unzählige Varianten denkbar. Algorithmus 3.1 zeigt das Grundgerüst eines Greedy-Algorithmus für die hier behandelte Problemstellung. Es sei an dieser Stelle erwähnt, dass durch die Existenz äquivalenter Codes mit systematischer Generatormatrix nach Satz 2.1 die Menge der betrachteten Spalten aus $\overline{\mathcal{V}_\lambda}$ in den ersten Schritten auf diejenigen beschränkt werden kann, die der systematischen Form genügen. Dieser Aspekt wurde bei der Implementierung des Programmes *Heurico* (vgl. Abschnitt 6.1) berücksichtigt, wir werden ihn hier jedoch der einfacheren Darstellung halber nicht weiter beachten.

Die Bewertungsfunktion wollen wir so konstruieren, dass ihr Wert $\mathcal{E}(x)$ als Schätzung für die Wahrscheinlichkeit angesehen werden kann, dass ein zufällig gewählter Vektor $x' \geq x$ mit $\mathbb{1}^T x' = n$ eine Lösung des Systems bildet.[3] Bevor wir uns jedoch ihrer

[1] beziehungsweise $x = \xi$, wobei ξ vom Benutzer vorgegeben wird, vgl. Algorithmus 3.1

[2] Ein Code heißt *constant-weight code*, wenn die in ihm enthaltenen Worte alle dasselbe Gewicht besitzen.

[3] Für zwei Vektoren $u = (u_0, u_1, \ldots, u_{l-1})$ und $v = (v_0, v_1, \ldots, v_{l-1})$ schreiben wir $u \geq v$ beziehungsweise $v \leq u$, wenn $u_i \geq v_i \; \forall i \in \{0, 1, \ldots, l-1\}$. Falls $u \geq v$ und ein i mit $u_i > v_i$ existiert, wird dies gelegentlich präzisierend mit $u > v$ beziehungsweise $v < u$ notiert werden.

Eingabe : **chainring** R: Grundring des zu konstruierenden Codes C
int $n, k, \lambda_0, \lambda_1, \ldots, \lambda_{k-1}, d$: Zielparameter für C
weightfunction w: Gewichtsfunktion
vector $\xi \in \mathbb{N}^{\mathfrak{v}}$: vorgegebener Startvektor mit $\mathbb{1}^T\xi < n$
Ausgabe : **failed** oder **vector** $x \in \mathbb{N}^{\mathfrak{v}}$ mit $x > \xi$, $\mathbb{1}^T x = n$ und $\mathcal{M}^w x \geq d \cdot \mathbb{1}$

```
1  procedure baseAlgorithm(...) : vector bzw. failed
2  {
3     vector x ← ξ;
4     while 𝟙^T x < n do
5        v* ←⊥; // ⊥ steht für undefiniert
6        foreach v ∈ V_λ do
7           if v* = ⊥ or E(x + e_v) > E(x + e_{v*}) then
8              v* ← v;
9           end if
10       end foreach
11       x ← x + e_{v*}; // e_{v*} ist Einheitsvektor mit 1 an Position v*
12    end while
13    if M^w x ≥ d · 𝟙 then
14       return x;
15    end if
16    return failed;
17 }
```

Algorithmus 3.1: Grundgerüst eines Greedy-Ansatzes

genaueren Definition widmen, soll zunächst die Struktur der Zeilen von $\mathcal{M}^w$ untersucht werden:

3.1. Zeilenstruktur von $\mathcal{M}^w$

Definition 3.1. Für $\overline{u} \in \overline{\mathcal{U}_\lambda}$ nennen wir das Tupel $(\tau_0^{\overline{u}}, \tau_1^{\overline{u}}, \ldots, \tau_m^{\overline{u}})$ mit

$$\tau_i^{\overline{u}} := \left|\{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(\overline{u}v) = i\}\right|$$

die *Höhenverteilung* der zu $\overline{u}$ gehörigen Zeile von $\mathcal{M}^w$. Für $t := \max\{w(r) : r \in R\}$ heißt das Tupel $(\omega_0^{\overline{u}}, \omega_1^{\overline{u}}, \ldots, \omega_t^{\overline{u}})$ mit

$$\omega_i^{\overline{u}} := \left|\{v \in \overline{\mathcal{V}_\lambda} : w(\overline{u}v) = i\}\right|$$

entsprechend die *Gewichtsverteilung* von $\overline{u}$.

Lemma 3.1. *Ist R ein Körper, so gilt $\tau_0^{\overline{u}} = q^{k-1}$ und $\tau_1^{\overline{u}} = \frac{q^{k-1}-1}{q-1}$ für alle $\overline{u} \in \overline{\mathcal{U}_\lambda}$.*

Beweis. Für $\overline{u} \in \overline{\mathcal{U}_\lambda}$ ist insbesondere $u \neq 0$ und daher $\left|u^\perp\right| = q^{k-1}$. Für $v \in u^\perp$ gilt, da R als Körper Kettenlänge $m = 1$ besitzt, $\mathfrak{h}(\overline{u}v) = \mathfrak{h}(0) = 1$. Abgesehen vom Nullvektor liegen je $q-1$ dieser Vektoren in einer Bahn unter Multiplikation mit Einheiten und somit folgt $\tau_1^{\overline{u}} = \frac{q^{k-1}-1}{q-1}$. Für die verbleibenden Vektoren $v \in \mathcal{V}_\lambda \setminus u^\perp$ gilt $\mathfrak{h}(\overline{u}v) = 0$. Hiervon gibt es $q^k - q^{k-1} = q^{k-1}(q-1)$ Stück und da wiederum je $(q-1)$ eine Bahn bilden, ergibt sich $\tau_0^{\overline{u}} = q^{k-1}$.

□

Die Berechnung der Höhenverteilungen der Zeilen von $\mathcal{M}^w$ im allgemeinen Fall $m > 1$ ist etwas komplizierter. Dennoch ist es nicht notwendig, dies direkt nach Definition 3.1 mittels Durchlaufen aller Spaltenrepräsentanten aus $\overline{\mathcal{V}_\lambda}$ zu tun. Schneller geht es, indem man die Beiträge der verschiedenen Repräsentantenklassen $\overline{\mathcal{V}_\lambda}(\vartheta, \zeta)$ (vgl. Definition 2.12) bestimmt und aufsummiert:

Lemma 3.2. *Sei* $\overline{u} = (\overline{u_0}, \overline{u_1}, \ldots, \overline{u_{k-1}}) \in \overline{\mathcal{U}_\lambda}$ *mit* $h_i := \mathfrak{h}(u_i)$ *und* ϑ *und* ζ *fest mit* $\vartheta \geq m - \lambda_\zeta$. *Dann gilt, mit* $\tau_i^{\vartheta,\zeta} := \left|\{v \in \overline{\mathcal{V}_\lambda}(\vartheta, \zeta) : \mathfrak{h}(\overline{u}v) = i\}\right|$ *und*
$\nu := \min\left(\{m\} \cup \{h_j + \vartheta + 1 : j \leq \zeta - 1\} \cup \{h_j + \max\{\vartheta, m - \lambda_j\} : \zeta + 1 \leq j \leq k-1\}\right)$*:*

(i) $s := \log_q(\left|\overline{\mathcal{V}_\lambda}(\vartheta, \zeta)\right|) = (m - (\vartheta+1))\zeta + \sum_{j=\zeta+1}^{k-1} \min\{m - \vartheta, \lambda_j\}$.

(ii) Falls $\nu > h_\zeta + \vartheta$, *so ist* $\tau_i^{\vartheta,\zeta} = 0$ *für* $i \neq h_\zeta + \vartheta$ *und* $\tau_{h_\zeta+\vartheta}^{\vartheta,\zeta} = q^s$.

(iii) Ist dagegen $\nu \leq h_\zeta + \vartheta$, *so gilt:*

$$\tau_i^{\vartheta,\zeta} = 0 \text{ für } i < \nu \ , \ \tau_m^{\vartheta,\zeta} = \frac{q^s}{q^{m-\nu}} \text{ und } \tau_i^{\vartheta,\zeta} = \frac{q^s}{q^{m-\nu}}(q^{m-i} - q^{m-i-1}) \text{ für } \nu \leq i < m.$$

Beweis.

(i) $\overline{\mathcal{V}_\lambda}(\vartheta, \zeta)$ besteht aus allen Vektoren mit Einträgen aus $\mathrm{Rad}(R)^{\vartheta+1}$ an den Stellen $0, \ldots, \zeta - 1$, gefolgt von der normierten Komponente $v_\zeta = \theta^\vartheta$ und Werten aus dem Ideal $\mathrm{Rad}(R)^{\max\{\vartheta, m-\lambda_i\}}$ an den Positionen $i > \zeta$. Mit $|\mathrm{Rad}(R)^i| = q^{m-i}$ folgt die Behauptung.

(ii) Für beliebiges $v = (v_0, v_1, \ldots, v_{k-1}) \in \overline{\mathcal{V}_\lambda}(\vartheta, \zeta)$ ist das Produkt $u_\zeta v_\zeta$ ein Element der Höhe $h_\zeta + \vartheta$ in R, während $u_i v_i$ für $i \neq \zeta$ mindestens Höhe ν besitzt. Gilt $\nu > h_\zeta + \vartheta$, so hat

$$\overline{u}v = \sum_{i=0}^{k-1} u_i v_i$$

unabhängig von v stets Höhe $h_\zeta + \vartheta$.[4] Damit ist die Behauptung klar.

[4] vgl. Lemma 1.4

(iii) Im Fall $\nu \leq h_\zeta + \vartheta$ betrachten wir die Menge $\widetilde{\mathcal{V}_\lambda} \subset R^{k-1}$ mit

$$\widetilde{\mathcal{V}_\lambda} := \left\{ (v_0, \ldots, v_{\zeta-1}, v_{\zeta+1}, \ldots, v_{k-1})^T : v_i \in \begin{cases} \mathrm{Rad}(R)^{\vartheta+1} & \text{für } i \leq \zeta - 1 \\ \mathrm{Rad}(R)^{\max\{m-\lambda_i, \vartheta\}} & \text{für } i \geq \zeta + 1 \end{cases} \right\}.$$

$\widetilde{\mathcal{V}_\lambda}$ ergibt sich aus $\overline{\mathcal{V}_\lambda}(\vartheta, \zeta)$ durch elementweises Streichen der fixierten ζ-ten Komponente $v_\zeta = \theta^\vartheta$. Die Abbildung

$$\phi : \widetilde{\mathcal{V}_\lambda} \to R, v = (v_0, \ldots, v_{\zeta-1}, v_{\zeta+1}, \ldots, v_{k-1})^T \mapsto \sum_{i \neq \zeta} u_i v_i$$

ist ein R-Rechtsmodulhomomorphismus, $I := \mathrm{Bild}(\phi)$ daher ein Ideal in R und die Mächtigkeit der Urbilder für jedes Element aus I gleich. Die minimale Höhe der Elemente in I ist ν und somit folgt $I = \mathrm{Rad}(R)^\nu$. Wegen $\nu \leq h_\zeta + \vartheta$ gilt $u_\zeta v_\zeta = u_\zeta \theta^\vartheta \in I$ und daher auch

$$\left\{ \sum_{i=0}^{k-1} u_i v_i : v \in \overline{\mathcal{V}_\lambda}(\vartheta, \zeta) \right\} = u_\zeta \theta^\vartheta + I = I.$$

Jeder Wert wird dabei genau $\frac{q^s}{|I|} = \frac{q^s}{q^{m-\nu}}$ mal angenommen. Weil es, für $i < m$, genau $q^{m-i} - q^{m-i-1}$ Elemente der Höhe i gibt, erhält man die angegebenen Werte für die $\tau_i^{\vartheta,\zeta}$.

□

3.2. Die Bewertungsfunktion $\mathcal{E}$

Definition 3.2. Für das zu Beginn des Kapitels aufgeführte Ungleichungssystem sowie $\overline{u} \in \overline{\mathcal{U}_\lambda}$, $t := \max\{w(r) : r \in R\}$ und $x \in \mathbb{N}^\mathfrak{v}$ mit $\mathbb{1}^T x \leq n$ sei:

(i) $\nu^{\overline{u}}(x) = (\nu_0^{\overline{u}}(x), \nu_1^{\overline{u}}(x), \ldots, \nu_t^{\overline{u}}(x))$ mit

$$\nu_i^{\overline{u}}(x) := \sum_{v \in \overline{\mathcal{V}_\lambda} : \mathcal{M}_{\overline{u}v}^w = i} x_v$$

die *Gewichtsverteilung* von x in Zeile $\overline{u}$.

(ii) $\sigma_{\overline{u}}(x) := \sum_{i=0}^{t} i \cdot \nu_i^{\overline{u}}(x) = \mathcal{M}_{\overline{u}*}^w x$ die *Gewichtssumme* von x für Zeile $\overline{u}$.

(iii) $\mathcal{N}(x) := \{x' \geq x : x' \in \mathbb{N}^\mathfrak{v}, \mathbb{1}^T x' = n\}$. Diese Menge wollen wir als die Menge der *zulässigen Nachfolger* von x bezeichnen.

(iv) $\mathcal{S}_{\overline{u}}(x) := \{x' \in \mathcal{N}(x) : \sigma_{\overline{u}}(x') \geq d\}$, die *Lösungsmenge zu x* für Zeile $\overline{u}$. Für $U \subset \overline{\mathcal{U}_\lambda}$ sei dann $\mathcal{S}_U(x)$ definiert durch

$$\mathcal{S}_U(x) := \bigcap_{\overline{u} \in U} \mathcal{S}_{\overline{u}}.$$

Statt $\mathcal{S}_{\overline{\mathcal{U}_\lambda}}$ schreiben wir einfach $\mathcal{S}(x)$ und nennen dies die *Lösungsmenge* zu x.

(v) Die Zahl $\delta_{\overline{u}}(x) := \frac{|\mathcal{S}_{\overline{u}}(x)|}{|\mathcal{N}(x)|}$ nennen wir die *Lösungsdichte zu x* für Zeile $\overline{u}$ und den Wert $\delta(x) := \frac{|\mathcal{S}(x)|}{|\mathcal{N}(x)|}$ einfach nur die *Lösungsdichte zu x*.

Dass eine exakte Berechnung von $\delta(x)$ im Allgemeinen schwer ist, lässt sich leicht einsehen, schließlich ist die Existenz einer Lösung des Ungleichungssystems äquivalent zu $\delta(\vec{0}) \neq 0$. Wir müssen uns für die Definition von $\mathcal{E}$ daher mit einer Schätzung zufrieden geben: Mit der Annahme, dass für disjunkte Mengen $U_1, U_2 \subset \overline{\mathcal{U}_\lambda}$ und zufällig gewähltes $x' \in \mathcal{N}(x)$ die Ereignisse „$x' \in \mathcal{S}_{U_1}$“ und „$x' \in \mathcal{S}_{U_2}$“ stochastisch unabhängig sind, erhalten wir die Näherung

$$\frac{|\mathcal{S}_U(x)|}{|\mathcal{N}(x)|} \approx \prod_{\overline{u} \in U} \frac{|\mathcal{S}_{\overline{u}}(x)|}{|\mathcal{N}(x)|} = \prod_{\overline{u} \in U} \delta_{\overline{u}}(x).$$

Das motiviert die Setzung

$$\mathcal{E}(x) := \prod_{\overline{u} \in \overline{\mathcal{U}_\lambda}} \delta_{\overline{u}}(x) \approx \delta(x).$$

$\delta_{\overline{u}}(x)$ hängt nur von $\overline{u}$ sowie $\mathbb{1}^T x$ und $\sigma_{\overline{u}}(x)$ ab, denn es gilt:

$$\delta_{\overline{u}}(x) = \frac{\left|\{x' \in \mathbb{N}^{\mathfrak{v}} : \mathbb{1}^T x' = n - \mathbb{1}^T x,\, \sigma_{\overline{u}}(x') \geq d - \sigma_{\overline{u}}(x)\}\right|}{|\{x' \in \mathbb{N}^{\mathfrak{v}} : \mathbb{1}^T x' = n - \mathbb{1}^T x\}|}.$$

Dies ist sofort einsichtig unter der Feststellung, dass die Mengen in Zähler und Nenner auf der rechten Seite der Gleichung durch die Zuordnung $x' \mapsto x + x'$ in Bijektion mit $\mathcal{S}_{\overline{u}}(x)$ bzw. $\mathcal{N}(x)$ stehen, also den Mengen in Zähler und Nenner von $\delta_{\overline{u}}(x)$.

Setzen wir

$$\epsilon_{\overline{u}} : \mathbb{N} \times \mathbb{Z} \to \mathbb{R}_0^+, (n', d') \mapsto \frac{\left|\{x' \in \mathbb{N}^{\mathfrak{v}} : \mathbb{1}^T x' = n',\, \sigma_{\overline{u}}(x') \geq d'\}\right|}{|\{x' \in \mathbb{N}^{\mathfrak{v}} : \mathbb{1}^T x' = n'\}|},$$

so gilt daher:

$$\delta_{\overline{u}}(x) = \epsilon_{\overline{u}}(n - \mathbb{1}^T x, d - \sigma_{\overline{u}}(x)).$$

Des Weiteren sind alle Einträge in $\mathcal{M}^w$ nichtnegativ, somit folgt $\epsilon_{\overline{u}}(n', d') = \epsilon_{\overline{u}}(n', 0) = 1$ für $d' \leq 0$. Während des Programmlaufs[5] gilt stets $0 \leq \mathbb{1}^T x \leq n$ und $\sigma_{\overline{u}}(x) \geq 0$, daher werden von $\epsilon_{\overline{u}}(n', d')$ nur die Werte auf der Menge $\{0, 1, \ldots, n\} \times \{0, 1, \ldots, d\}$ benötigt. Diese können zu Beginn in einer Tabelle vorberechnet werden. Weil $\epsilon_{\overline{u}}$ nicht von $\overline{u}$ selbst, sondern nur von der Gewichtsverteilung[6] von $\overline{u}$ abhängt, muss diese Berechnung für alle Zeilen mit gleicher Gewichtsverteilung nur einmal erfolgen. Insbesondere fallen hierbei also alle Zeilen mit gleicher Höhenverteilung zusammen.

[5] vgl. Algorithmus 3.1
[6] vgl. Definition 3.1

3.3. Vorberechnung von $\epsilon_{\overline{u}}$ in einer Tabelle

Eine mögliche Implementierung dieser Vorberechnung ist in Algorithmus 3.2 dargestellt. Darin werden rekursiv alle möglichen Gewichtsverteilungen für Vektoren x' mit $\mathbb{1}^T x' \leq n$ in Zeile $\overline{u}$ durchlaufen[7] und in Zeile 20 die Anzahl dieser Vektoren auf den zu $\mathbb{1}^T x'$ und $\sigma_{\overline{u}}(x')$ gehörenden Tabelleneintrag addiert. Das zum Schluss durchgeführte Aufsummieren in Zeile 10 sorgt dafür, dass $\epsilon_{\overline{u}}[n'][d']$ tatsächlich die Zahl der Vektoren x' mit $\mathbb{1}^T x' = n'$ und $\sigma_{\overline{u}}(x') \geq d'$ enthält. Daher gilt am Ende

$$\epsilon_{\overline{u}}[n'][0] = \left|\{x' : \mathbb{1}^T x' = n'\}\right| = \binom{n' + \mathfrak{v} - 1}{n'}$$

und Zeile 13 sorgt für die richtige Normierung.[8] Die Zahl der rekursiven Aufrufe von *recursiveComputation* ist gerade die Anzahl c verschiedener Gewichtsverteilungen unter den Vektoren x' mit $\mathbb{1}^T x' \leq n$. Ist $(\omega_0^{\overline{u}}, \omega_1^{\overline{u}}, \ldots, \omega_t^{\overline{u}})$ die Gewichtsverteilung von Zeile $\overline{u}$, so hängt c lediglich von n und der Anzahl l der von Null verschiedenen Komponenten $\omega_i^{\overline{u}}$ ab und berechnet sich durch

$$c = \sum_{i=0}^{n} \binom{i+l-1}{i} = \sum_{i=0}^{n} \binom{i+l-1}{l-1}.$$

Bei den in dieser Arbeit durchgeführten Suchen nach linearen Codes über Kettenringen liegt n typischerweise in der Größenordnung $n \in [1, 100]$. Für unsere Zwecke ist Algorithmus 3.2 also für kleine Werte von l (etwa $l \in [2, 5]$), wie sie etwa bei Verwendung des Hamminggewichts ($l = 2$) oder des homogenen Gewichts ($l = 3$) auftreten, in praktikabler Zeit durchführbar.

Eine Alternative besteht im folgenden Ansatz: Berechnet man die Koeffizienten des Polynoms

$$p_{\overline{u}}(\xi, \eta) := \prod_{i:\omega_i^{\overline{u}} > 0} \left(\sum_{j=0}^{n} \xi^j \eta^{ij} \binom{j + \omega_i^{\overline{u}} - 1}{j} \right),$$

so gibt der Koeffizient von $\xi^{n'}\eta^{d'}$ gerade die Anzahl verschiedener Vektoren x' mit $\mathbb{1}^T x' = n'$ und $\sigma_{\overline{u}}(x') = d'$ an. Durch entsprechendes Aufsummieren und anschließendes Normieren erhält man hieraus die Werte für $\epsilon_{\overline{u}}$. Beim Ausmultiplizieren der Faktoren von $p_{\overline{u}}(\xi, \eta)$ können Monome, bei denen der Exponent von ξ größer ist als n, weggelassen werden; ebenso können alle Monome $\xi^j \eta^i$ mit $i \geq d$ zusammengefasst werden in einem Monom $\xi^j \eta^d$. In jedem Schritt der Ausmultiplikation besitzt das so reduzierte Polynom also maximal $(n+1)\cdot(d+1)$ Koeffizienten, die mit den $(n+1)$ verschiedenen des nächsten Faktors multipliziert werden müssen. Die Gesamtzahl μ nötiger Multiplikationen lässt sich somit abschätzen durch $\mu \leq l \cdot (n+1)(d+1) \cdot (n+1) = l(d+1)(n+1)^2$. Bei praxisnahen Problemen ist der durch die Vorberechnung der Tabellen entstehende Aufwand damit einigermaßen vernachlässigbar.

[7] Die Gewichtsverteilungen werden repräsentiert durch das Array $\nu[0..t]$, wobei $\nu[i]$ angibt, wie oft Gewicht i in x' vorkommt.

[8] $\binom{n+k-1}{n}$ ist die kombinatorische Anzahl möglicher Verteilungen von n gleichartigen Objekten auf k Plätze, wobei auf jeden Platz beliebig viele Objekte entfallen dürfen.

3.4. $\mathcal{E}$ und Grundalgorithmus im Beispiel

Beispiel 3.1. Sei wieder $R = \mathbb{Z}_9$, $k = 2$, $\lambda_0 = 2$, $\lambda_1 = 1$, $n = 3$, $w = w_{\text{hom}}$ und $d = 6$. Die Mengen $\overline{\mathcal{V}_\lambda}$ und $\overline{\mathcal{U}_\lambda}$ wurden bereits in Beispiel 2.2 aufgelistet. Bei Indizierung der Zeilen und Spalten in der dort verwendeten Reihenfolge hat die Matrix $\mathcal{M}^w$ die Gestalt

$$\mathcal{M}^w = \left(\begin{array}{ccccccc} 2 & 2 & 2 & 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 3 & 3 & 0 & 3 \\ 2 & 2 & 2 & 3 & 0 & 3 & 3 \\ \hline 3 & 3 & 3 & 0 & 0 & 0 & 0 \\ \hline 3 & 3 & 0 & 0 & 3 & 3 & 3 \\ 3 & 0 & 3 & 0 & 3 & 3 & 3 \\ 0 & 3 & 3 & 0 & 3 & 3 & 3 \end{array}\right).$$

Es bezeichne ω_i^j die Vielfachheit, mit der Gewicht i in Zeile j ($j = 0, \ldots, 6$) auftritt. Die ersten und die letzten drei Zeilen haben jeweils die gleiche Gewichtsverteilung, daher gilt $\omega_i^0 = \omega_i^1 = \omega_i^2$ und $\omega_i^4 = \omega_i^5 = \omega_i^6$ für $i = 0, \ldots, 3$. Die verschiedenen Verteilungen sind:

j	ω_0^j	ω_1^j	ω_2^j	ω_3^j
0	1	0	3	3
3	4	0	0	3
4	2	0	0	5

Das Polynom $p_0(\xi, \eta)$ berechnet sich durch

$$\begin{aligned} p_0(\xi,\eta) = {} & (1 + \xi^1 + \xi^2 + \xi^3) \cdot (1 + 3\xi^1\eta^2 + 6\xi^2\eta^4 + 10\xi^3\eta^6) \cdot \\ & \cdot (1 + 3\xi^1\eta^3 + 6\xi^2\eta^6 + 10\xi^3\eta^9). \end{aligned}$$

Nach Reduktion modulo ξ^4 und Zusammenfassen aller Terme $\xi^j\eta^i$ mit $i \geq 6$ zu $\xi^j\eta^6$ liefert Ausmultiplizieren

$$\begin{aligned} \overline{p_0}(\xi,\eta) = {} & \xi^3(62\eta^6 + 9\eta^5 + 6\eta^4 + 3\eta^3 + 3\eta^2 + 1) + \\ & + \xi^2(6\eta^6 + 9\eta^5 + 6\eta^4 + 3\eta^3 + 3\eta^2 + 1) + \\ & + \xi^1(3\eta^3 + 3\eta^2 + 1) + 1. \end{aligned}$$

Das ergibt die folgende Wertetabelle für ϵ_0:

$$\epsilon_0(n', d') = \begin{array}{|c||c|c|c|c|c|c|c|} \hline n'/d' & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline\hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & \frac{6}{7} & \frac{6}{7} & \frac{3}{7} & 0 & 0 & 0 \\ \hline 2 & 1 & \frac{27}{28} & \frac{27}{28} & \frac{24}{28} & \frac{21}{28} & \frac{15}{28} & \frac{6}{28} \\ \hline 3 & 1 & \frac{83}{84} & \frac{83}{84} & \frac{80}{84} & \frac{77}{84} & \frac{71}{84} & \frac{62}{84} \\ \hline \end{array}$$

Analog erhält man für ϵ_3 beziehungsweise ϵ_4:

$\epsilon_3(n', d') =$

n'/d'	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0
1	1	$\frac{3}{7}$	$\frac{3}{7}$	$\frac{3}{7}$	0	0	0
2	1	$\frac{18}{28}$	$\frac{18}{28}$	$\frac{18}{28}$	$\frac{6}{28}$	$\frac{6}{28}$	$\frac{6}{28}$
3	1	$\frac{64}{84}$	$\frac{64}{84}$	$\frac{64}{84}$	$\frac{34}{84}$	$\frac{34}{84}$	$\frac{34}{84}$

$\epsilon_4(n', d') =$

n'/d'	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0
1	1	$\frac{5}{7}$	$\frac{5}{7}$	$\frac{5}{7}$	0	0	0
2	1	$\frac{25}{28}$	$\frac{25}{28}$	$\frac{25}{28}$	$\frac{15}{28}$	$\frac{15}{28}$	$\frac{15}{28}$
3	1	$\frac{80}{84}$	$\frac{80}{84}$	$\frac{80}{84}$	$\frac{65}{84}$	$\frac{65}{84}$	$\frac{65}{84}$

Wir betrachten nun die Anwendung von Algorithmus 3.1 auf diese Situation, wobei wir für ξ den Nullvektor vorgeben. $x^{(i)}$ bezeichne den Vektor x im i-ten Schritt (also wenn $\mathbb{1}^T x = i$). Weiter sei $\Delta_j^i := \mathcal{E}(x^{(i)} + e_j)$. Dann wird der Ablauf durch folgende Tabelle beschrieben:

i	$x^{(i)}$	Δ_0^i	Δ_1^i	Δ_2^i	Δ_3^i	Δ_4^i	Δ_5^i	Δ_6^i
0	$\vec{0}$	**0.116**	0.116	0.116	0.021	0.024	0.024	0.024
1	e_0	0	**0.321**	0.321	0	0	0	0
2	$e_0 + e_1$	0	0	**1**	0	0	0	0
3	$e_0 + e_1 + e_2$	–	–	–	–	–	–	–

Ist der Eintrag Δ_j^i fett gedruckt, so bedeutet dies, dass $x^{(i)}$ im i-ten Schritt um den Vektor e_j inkrementiert wird. Die von 0 beziehungsweise 1 verschiedenen Werte der Δ_j^i kommen so zustande:

$$\begin{aligned}
\Delta_0^0 = \Delta_1^0 = \Delta_2^0 &= \epsilon_0(2,4)^3 \cdot \epsilon_3(2,3)^1 \cdot \epsilon_4(2,3)^2 \cdot \epsilon_4(2,6)^1 = \tfrac{2278125}{19668992} \approx 0.116 \\
\Delta_3^0 &= \epsilon_0(2,3)^3 \cdot \epsilon_3(2,6)^1 \cdot \epsilon_4(2,6)^3 = \tfrac{273375}{13176688} \approx 0.021 \\
\Delta_4^0 = \Delta_5^0 = \Delta_6^0 &= \epsilon_0(2,3)^2 \cdot \epsilon_0(2,6)^1 \cdot \epsilon_3(2,6)^1 \cdot \epsilon_4(2,3)^3 = \tfrac{1265625}{52706752} \approx 0.024 \\
\Delta_1^1 = \Delta_2^1 &= \epsilon_0(1,2)^3 \cdot \epsilon_3(1,0)^1 \cdot \epsilon_4(1,0)^1 \cdot \epsilon_4(1,3)^2 = \tfrac{5400}{16807} \approx 0.321
\end{aligned}$$

Das Ergebnis von Algorithmus 3.1 ist also der Vektor $x^{(3)} = (1,1,1,0,0,0,0)^T$ und entspricht der Generatormatrix

$$\Gamma = \begin{pmatrix} \overline{1} & \overline{1} & \overline{1} \\ \overline{0} & \overline{3} & \overline{6} \end{pmatrix}.$$

Summiert man die ersten drei Spalten von $\mathcal{M}^w$ auf und beachtet, dass jedes $\overline{u} \in \overline{\mathcal{U}_\lambda}$ stellvertretend für $|R^\times(\overline{u})|$ Codeworte steht, erkennt man, dass der von Γ erzeugte $\mathbb{Z}_9$-lineare Code aus dem Nullwort sowie 24 Codeworten von Gewicht 6 und 2 Codeworten von Gewicht 9 besteht.

3.5. Backtracking-Algorithmus

Ein unbefriedigender Aspekt von Algorithmus 3.1 ist, dass die Suche direkt nach Erreichen von $\mathbb{1}^T x = n$ als gescheitert abgebrochen wird, falls x keine Lösung darstellt. Daher bietet es sich an, den Algorithmus um einen Backtracking-Mechanismus zu erweitern. Eine mögliche Variante soll hier vorgestellt werden:

Das Prinzip eines Backtracking-Ansatzes ist in diesem Fall, einige der zuletzt durchgeführten Inkrementierungen rückgängig zu machen und danach die Suche durch Erhöhung einer anderen Variable als zuvor wieder aufzunehmen. Die Werte der Funktion $\mathcal{E}$ liefern hierfür einen guten Ausgangspunkt, erlauben sie es doch, für jeden Schritt eine (fast) totale Ordnung unter allen wählbaren Variablen herzustellen. Die Einschränkung ergibt sich dadurch, dass verschiedene Variablen in einem Schritt durchaus dieselbe Bewertung erhalten können, wie schon das Beispiel 3.1 zeigt. Konkret sieht unser Entwurf wie folgt aus:

Sei $a \geq 1.0$ ein vom Benutzer bei Programmstart gewählter Wert. Im i-ten Schritt (d. h. $\mathbb{1}^T x = i$) wird nun zunächst eine Liste L_i berechnet, die die Paare aus den Variablen und ihren Bewertungen enthält. Diese Liste wird anschließend absteigend nach den Bewertungen sortiert und die höchste Bewertung in die Variable β_i^* geschrieben. Falls $\beta_i^* = 0$, kann die Suche in diesem Teilbaum sofort abgebrochen werden, andernfalls wird solange über die Paare (v_i, β_i) der Liste iteriert, bis entweder eine Lösung gefunden wurde oder $a \cdot \beta_i < \beta_i^*$ gilt. Im zweiten Fall setzt das Backtracking ein: Ist die Kette der Vorgänger zum aktuellen Vektor $x^{(i)}$ gegeben durch

$$\xi = x^{(\mathbb{1}^T \xi)} < x^{(\mathbb{1}^T \xi + 1)} < \cdots < x^{(i-1)} < x^{(i)},$$

so wird die Variable i', beginnen mit $i' := i-1$, so lange dekrementiert, bis die zugehörige Liste $L_{i'}$ nicht leer ist. Aus $L_{i'}$ wird nun der oberste Eintrag $(v_{i'}, \beta_{i'})$ entfernt und die Suche anschließend mit dem Vektor $x^{(i'+1)} := x^{(i')} + e_{v_{i'}}$ zur Kette

$$\xi = x^{(\mathbb{1}^T \xi)} < x^{(\mathbb{1}^T \xi + 1)} < \cdots < x^{(i')} < x^{(i'+1)}$$

im Schritt $i = i' + 1$ fortgesetzt. In den nachfolgenden Schritten werden die Listen L_i für $i \geq i' + 1$ dabei jeweils neu berechnet und ihre alten Daten überschrieben. Erst wenn zum Zeitpunkt eines Backtracking-Versuchs alle Listen leer sind, wird die Suche gestoppt. Je größer a gewählt wird, umso mehr nähert sich der Algorithmus einer vollständigen brute-force-Suche (mit heuristisch bestimmter Suchreihenfolge) an.

Abbildung 3.1 zeigt den schematischen Ablauf des Backtrackings und Algorithmus 3.3 den Pseudocode.

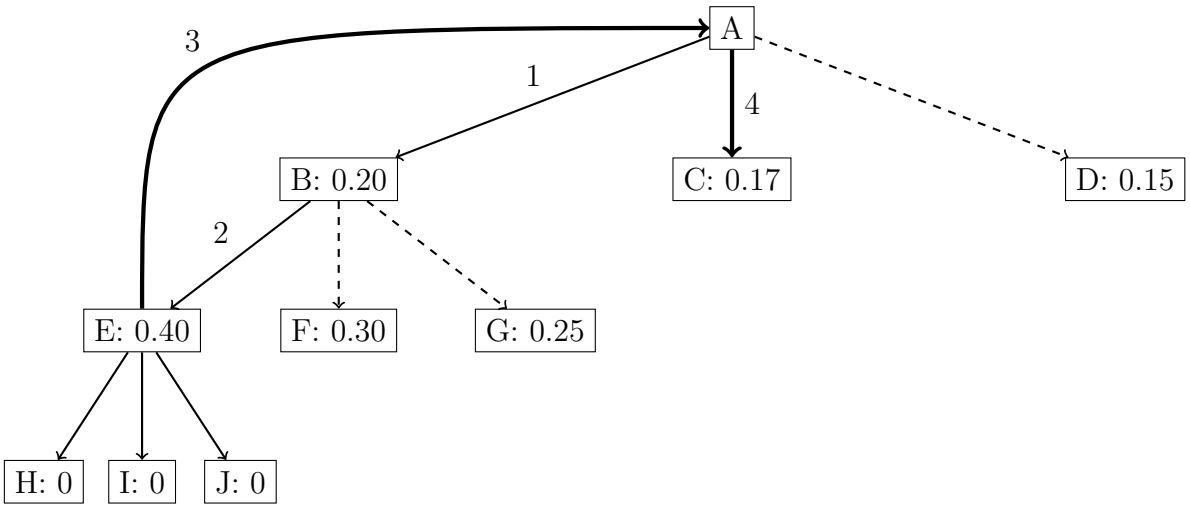

Abbildung 3.1.: Schematischer Ablauf von Algorithmus 3.3 zum Parameter $a = 1.25$: Die Kanten repräsentieren die wählbaren Variablen, deren Bewertungen im jeweiligen Zielknoten aufgeführt sind. Die Zahlen an den Kanten geben die Durchlaufreihenfolge an. Ausgehend vom Knoten A werden also zunächst zweimal die jeweils bestbewerteten Variablen inkrementiert. Im Knoten E wird erkannt, dass keine Lösung mehr möglich ist (alle Variablen haben Bewertung 0) und ein Backtracking-Schritt durchgeführt: Da $0.25 \cdot 1.25 < 0.30 \cdot 1.25 < 0.4$, bleiben die Knoten F und G unberücksichtigt und die Suche nach einer Alternative wird im Vater von B, also dem Knoten A, durchgeführt. Die Variable zur zweiten Kante ist eine solche, denn $0.17 \cdot 1.25 \geq 0.2$; der Algorithmus wird daher im Knoten C fortgesetzt.

3.6. Implementierungsdetails

3.6.1. Inkrementelles Update der Gewichtssummen

Da der Vektor $x^{(i)}$ zwischen den rekursiven Aufrufen von *backtrackAlg* in Algorithmus 3.3 bzw. bei den Aufrufen von $\mathcal{E}$ immer nur an einer Koordinate modifiziert wird, ist es umständlich, die neuen Gewichtssummen jedes Mal gemäß Definition 3.2 von Grund auf neu zu berechnen. Stattdessen sollte jeweils nur die Veränderung ermittelt und die Werte entsprechend aktualisiert werden. In diesem Fall muss in jedem Vorwärts- und Rückwärtsschritt und bei Auswerten von $\mathcal{E}$ nur auf eine Spalte von $\mathcal{M}^w$ zugegriffen werden.

3.6.2. Temporäre Entfernung von Zeilen und Spalten

Die Definition von $\mathcal{E}$ erlaubt den Einsatz zweier Tricks, die die Durchführung von Algorithmus 3.1 beziehungsweise 3.3 erheblich beschleunigen können:

Lemma 3.3. *Für $x \in \mathbb{N}^{\mathfrak{v}}$ mit $\mathbb{1}^T x < n$ gilt:*

$$[\mathcal{E}(x + e_j) = 0] \Rightarrow \mathcal{E}(x' + e_j) = 0 \; \forall x' \in \mathbb{N}^{\mathfrak{v}} \text{ mit } x' \geq x,\ \mathbb{1}^T x' < n.$$

Beweis. Die Gleichung

$$\mathcal{E}(x + e_j) = \prod_{\overline{u} \in \overline{\mathcal{U}_\lambda}} \delta_{\overline{u}}(x + e_j) = 0$$

impliziert die Existenz eines $\overline{u}^* \in \overline{\mathcal{U}_\lambda}$ mit $\delta_{\overline{u}^*}(x+e_j) = 0$, was wiederum äquivalent ist zu $\mathcal{S}_{\overline{u}^*}(x + e_j) = \emptyset$. Für $x' \geq x$ mit $\mathbb{1}^T x' < n$ folgt dann, wegen $\mathcal{S}_{\overline{u}^*}(x' + e_j) \subset \mathcal{S}_{\overline{u}^*}(x + e_j)$: $\mathcal{S}_{\overline{u}^*}(x' + e_j) = \emptyset$, also $\mathcal{E}(x' + e_j) = 0$.

□

Lemma 3.4. *Für $x \in \mathbb{N}^{\mathfrak{v}}$ mit $\mathbb{1}^T x < n$ gilt:*

$$[\delta_{\overline{u}}(x) = 1] \Rightarrow \delta_{\overline{u}}(x') = 1 \; \forall x' \in \mathbb{N}^{\mathfrak{v}} \text{ mit } x' \geq x,\ \mathbb{1}^T x' \leq n.$$

Beweis. Aus $\delta_{\overline{u}}(x) = 1$ folgt $\mathcal{S}_{\overline{u}}(x) = \mathcal{N}(x)$. Aufgrund der Implikationskette

$$y \in \mathcal{N}(x') \overset{\mathcal{N}(x') \subseteq \mathcal{N}(x)}{\Rightarrow} [y \in \mathcal{N}(x) = \mathcal{S}_{\overline{u}}(x)] \Rightarrow [\sigma_{\overline{u}}(y) \geq d] \Rightarrow y \in \mathcal{S}_{\overline{u}}(x')$$

folgt weiter $\mathcal{N}(x') = \mathcal{S}_{\overline{u}}(x')$ und damit $\delta_{\overline{u}}(x') = 1$.

□

Lemma 3.3 erlaubt, bei der Wahl der nächsten zu inkrementierenden Variablen risikolos diejenigen auszuschließen, deren Bewertung schon in den vorangehenden Schritten Null war. Mithilfe von Lemma 3.4 dagegen lassen sich temporär solche Ungleichungen aus dem System entfernen, die ohnehin schon sicher erfüllt sind. Das entspricht der Situation, dass in einer Zeile $\overline{u}$ für den Vektor $x \in \mathbb{N}^{\mathfrak{v}}$ mit $\mathbb{1}^T x < n$ die Gewichtssumme $\sigma_{\overline{u}}(x)$ bereits $\geq d$ ist. Um maximal von dieser Beobachtung zu profitieren, empfiehlt es sich, die Ungleichungen beziehungsweise Variablen in einer doppelt verketteten Liste zu organisieren. Dies erlaubt einerseits, die Listen durchzugehen, ohne bereits entfernte Einträge in irgendeiner Form antasten zu müssen. Andererseits können diese dann nach einem Backtracking-Schritt auch schnell wieder an ihre alte Position in die Liste eingefügt werden. Sehr instruktiv wird dieses Verfahren beispielsweise in [Knu00] eingesetzt.

3.6.3. Ein zeilenübergreifendes Abbruchkriterium

Sei $x^{(i)}$ der Vektor im i-ten Schritt von Algorithmus 3.3 und $Z := \{\overline{u} \in \overline{\mathcal{U}_\lambda} : \sigma_{\overline{u}}(x^{(i)}) < d\}$. Wir betrachten die Werte

$$\Delta := \sum_{\overline{u} \in Z} d - \sigma_{\overline{u}}(x^{(i)}) \text{ und } \mu := \max_{v \in \overline{\mathcal{V}_\lambda}} \{\sum_{\overline{u} \in Z} m_{\overline{u}v}\},$$

wobei $m_{\overline{u}v}$ den Eintrag von $\mathcal{M}^w$ in Zeile $\overline{u}$ und Spalte v bezeichnet. Gilt $(n-i)\mu < \Delta$, so ist klar, dass in den verbleibenden $(n-i)$ Schritten die Gewichtssumme nicht mehr in allen Zeilen auf mindestens d gehoben werden kann. Demzufolge kann an dieser Stelle sofort ein Backtracking-Schritt durchgeführt werden. Das Kriterium greift besonders häufig dann, wenn schon in vielen Zeilen die Gewichtssumme größer als d ist, denn dies bedeutet gewissermaßen eine Verschwendung der verfügbaren „Gewichtsmasse“ (diese beträgt $n\,|C|\,(q-1)$, falls $w = w_{\text{hom}}$, vgl. Lemma 2.3).

3.6.4. Behandlung von Problemen mit der Rechengenauigkeit

Schon die Tabelleneinträge $\epsilon_{\overline{u}}(n', d')$ sind oft sehr klein und damit natürlich erst recht die Werte von $\mathcal{E}(x)$. Das kann, speziell bei großer Kardinalität von $\overline{\mathcal{U}_\lambda}$, zu Problemen mit der Rechengenauigkeit führen. Diese lassen sich aber weitestgehend beseitigen, wenn intern mit dem Logarithmus von $\mathcal{E}$ gerechnet wird:

$$\log(\mathcal{E}(x)) = \log(\prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \delta_{\overline{u}}(x)) = \sum_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \log(\delta_{\overline{u}}(x)) = \sum_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \log(\epsilon_{\overline{u}}(n - \mathbb{1}^T x, d - \sigma_{\overline{u}}(x))).$$

Beim Anlegen der vorberechneten Tabelle sollten dann entsprechend ebenfalls die logarithmierten Werte von $\epsilon_{\overline{u}}(n', d')$ gespeichert werden; für den Fall $\epsilon_{\overline{u}}(n', d') = 0$ müssen dabei natürlich geeignete Vorkehrungen getroffen werden. Weiterhin ist für die in Abschnitt 3.7.2 beschriebene heuristische Isometrieerkennung wichtig, dass beim rechnerinternen Aufsummieren dieser Werte die Kommutativität gilt. Das ist jedoch bei Verwendung von Gleitkommazahlen nicht immer gegeben. Abhilfe schafft es, wenn man beim Berechnen der logarithmierten Tabelle die Gleitkommawerte vor dem Ablegen noch mit einer großen Konstante c multipliziert[9] und das (gerundete) Ergebnis dann als ganze Zahl speichert.

Im Weiteren lassen wir diese Überlegungen aber außer Acht und tun so, als ob alle Gleitkommaoperationen exakt durchgeführt werden könnten. Daher bleiben wir, insbesondere in Kapitel 3.7.2, bei den ursprünglichen Definitionen von $\mathcal{E}$ und $\epsilon_{\overline{u}}$.

3.6.5. Cachefreundliche Speicherung von $\mathcal{M}^w$

Auf modernen Computersystemen werden häufig benötigte Teile des Arbeitsspeichers (RAM[10]) in einen speziellen Zwischenspeicher, den sogenannten *Cache*, kopiert. Zugriffe auf ihn erfolgen in der Regel um ein Vielfaches schneller als auf den RAM, der allerdings dafür deutlich größer ist.[11] Somit können immer nur Teile des RAM im Cache gelagert werden. Je häufiger eine Speicheranfrage des Prozessors auf einen im Cache befindlichen Bereich fällt, desto schneller läuft das jeweilige Programm. Daher versuchen moderne Mikroprozessoren mit Hilfe heuristischer Strategien, möglichst viele der innerhalb der

[9] Bei den Programmen *Heurico* und *Solver* wurde beispielsweise $c := 1.0 \cdot 10^{14}$ gesetzt.

[10] RAM = **R**andom **A**ccess **M**emory

[11] Auf derzeit üblichen Systemen liegt die Größe des RAM im Gigabytebereich, während die Cachegröße je nach Prozessor einige Megabyte beträgt.

nächsten Operationen angeforderten Speicherinhalte im Cache bereitzustellen. Eine dieser Heuristiken besteht darin, nach Zugriff auf eine Speicherzelle auch den Inhalt der darauf folgenden Zellen in den Cache zu nehmen - schließlich ist es durchaus wahrscheinlich, dass ein Algorithmus die Daten im Arbeitsspeicher sequentiell abarbeitet. Im Falle des hier beschriebenen Verfahrens muss, auch bei inkrementeller Berechnung der Gewichtssummen wie in Abschnitt 3.6.1, bei jeder Bewertung einer Spalte $v \in \overline{\mathcal{V}_\lambda}$ zumindest einmal die zugehörige Spalte von $\mathcal{M}^w$ durchlaufen zu werden, um die aktuellen Gewichtssummen der Zeilen $\overline{u} \in \overline{\mathcal{U}_\lambda}$ nach Anfügen von v an die Generatormatrix und damit die Indizes für das Nachschlagen in der Tabelle $\epsilon_{\overline{u}}$ zu ermitteln. Bei einer hinsichtlich der zuvor genannten Caching-Strategie günstigen Implementierung sollte daher die Matrix $\mathcal{M}^w$ *spaltenweise* im Arbeitsspeicher abgelegt werden, also so, dass die Einträge einer Spalte im Speicher direkt aufeinander folgen. Bei der Implementierung des Programmes *Heurico* konnte auf diese Weise mit einem zum Zeitpunkt der Verfassung dieser Arbeit handelsüblichen Prozessor[12] eine mittlere Geschwindigkeitssteigerung von rund 50 Prozent gegenüber zeilenweiser Speicherung von $\mathcal{M}^w$ erzielt werden.

3.7. Behandlung semilinear äquivalenter Generatormatrizen

Im ersten Teil dieses Abschnitts wollen wir diskutieren, ob semilinear äquivalente Generatormatrizen (vgl. Definition 2.24) auch im Kontext von Algorithmus 3.3 gleichwertig sind. Konkreter lautet die Frage, ob das Suchverhalten für Vektoren $x^{(i)}$ und $y^{(i)}$, deren zugehörige Generatormatrizen $\Gamma_{x^{(i)}}$ und $\Gamma_{y^{(i)}}$ semilinear äquivalent sind, im darunter liegenden Teilbaum identisch ist. Es wird sich herausstellen, dass die Frage uneingeschränkt mit „Ja“ beantwortet werden kann, womit sich ein großer Teil an Rechenzeit einsparen lässt: Wurde der Teilbaum unterhalb von $x^{(i)}$ schon untersucht, braucht derjenige zu $y^{(i)}$ nicht mehr betrachtet werden, da hier nichts wesentlich anderes passieren würde; dementsprechend kann sofort ein Backtracking-Schritt durchgeführt werden. Natürlich braucht man dazu eine Möglichkeit, die semilineare Äquivalenz von $\Gamma_{x^{(i)}}$ und $\Gamma_{y^{(i)}}$ festzustellen oder zumindest mit guter Trefferquote zu „raten“, ob diese vorliegt. Daher wollen wir im zweiten Teil eine Heuristik vorstellen, die eine solche Abschätzung mit relativ geringem Zusatzaufwand vornimmt.

3.7.1. Zur Identifizierbarkeit semilinearer Äquivalenzklassen im Algorithmus

Definition 3.3. Seien n, w, d und λ wie bisher und $n_0 < n$. Für eine Generatormatrix $\Gamma \in G^\lambda_{n_0}$ sei dann $\mathcal{E}(\Gamma) := \mathcal{E}(x_\Gamma)$.

[12] Intel Core2Quad Q6600

Bemerkung 3.1. Für $\Gamma = (\gamma_0|\gamma_1|\ldots|\gamma_{n_0-1})$ gilt

$$\mathcal{E}(\Gamma) = \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}}(n-n_0, d-\sum_{j=0}^{n_0-1} w(\overline{u}\gamma_j)).$$

Lemma 3.5. *Sei σ ein Ringautomorphismus von R und $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ ein Umriss. Für $A \in \mathcal{A}^\lambda$ ist die Abbildung*

$$\mathcal{L}_A^\sigma : \mathcal{V}_\lambda \to \mathcal{V}_\lambda, v \mapsto A\sigma(v)$$

ein semilinearer R-Rechtsmodulautomorphismus auf $\mathcal{V}_\lambda$ und $\overline{\mathcal{V}_\lambda}$ wird durch $\mathcal{L}_A^\sigma$ wieder auf ein Vertretersystem von ${\mathcal{V}_\lambda}^*$ unter Rechtsmultiplikation mit Einheiten abgebildet.

Beweis. Betrachtet man die Abbildung $v \mapsto A\sigma(v)$ auf ganz R^k, so handelt es sich wegen der Invertierbarkeit von A um einen semilinearen R-Rechtsmodulautomorphismus. Sowohl σ als auch die Linksmultiplikation mit A (vgl. Satz 2.22) bilden $\mathcal{V}_\lambda$ auf sich selbst ab, also ist $\mathcal{L}_A^\sigma$ eine wohldefinierte Einschränkung der Komposition und besitzt dieselben Struktureigenschaften. Da $\mathcal{L}_A^\sigma(vr) = \mathcal{L}_A^\sigma(v)\sigma(r)$ für beliebige $v \in {\mathcal{V}_\lambda}^*$ und $r \in R^\times$ gilt und $\sigma(r)$ wieder eine multiplikative Einheit in R ist, bildet $\mathcal{L}_A^\sigma$ Bahnen der Operation von $R^\times$ auf ${\mathcal{V}_\lambda}^*$ per Rechtsmultiplikation wieder auf ebensolche Bahnen ab. Damit folgt der zweite Teil der Behauptung.

□

Lemma 3.6. *Sei σ ein Ringautomorphismus von R, $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$ ein Umriss und $A \in \mathcal{A}^\lambda$. Die Abbildung*

$$\mathcal{R}_A^\sigma : \mathcal{U}_\lambda \to \mathcal{U}_\lambda, \overline{u} \mapsto \overline{\sigma(u)A}$$

ist ein semilinearer R-Linksmodulautomorphismus, der nur Elemente gleicher Höhenverteilung[13] permutiert. Außerdem wird $\overline{\mathcal{U}_\lambda}$ durch $\mathcal{R}_A^\sigma$ wieder auf ein Vertretersystem von ${\mathcal{U}_\lambda}^*$ unter Linksmultiplikation mit Einheiten abgebildet.

Beweis. Dass $\mathcal{R}_A^\sigma$ ein semilinearer R-Linksmodulautomorphismus auf $\mathcal{U}_\lambda$ ist, folgt sofort, nachdem die Wohldefiniertheit gezeigt ist. Seien also $x, y \in R^k$, so dass $\overline{x} = \overline{y} \in \mathcal{U}_\lambda$, etwa $y = x + \delta$ mit $\delta = (r_0\theta^{\lambda_0}, r_1\theta^{\lambda_1}, \ldots, r_{k-1}\theta^{\lambda_{k-1}})$ und $r_i \in R$ geeignet. Es folgt $\sigma(y)A = \sigma(x)A + \sigma(\delta)A$. Für $i \geq j$ gilt $a_{ij} \in \mathrm{Rad}(R)^{\lambda_j-\lambda_i}$ und σ erhält komponentenweise die Höhen, daher liegt die j-te Komponente von $\sigma(\delta)A$ in $\mathrm{Rad}(R)^{\lambda_j}$. Also ist $\overline{\sigma(y)A} = \overline{\sigma(x)A}$ und $\mathcal{R}_A^\sigma$ wohldefiniert. Für die Aussage zur Höhenverteilung schließt man zunächst mit Hilfe von Lemma 3.5, dass $\mathcal{L}_{A^{-1}}^\sigma$ ein semilinearer R-Rechtsmodulautomorphismus auf $\mathcal{V}_\lambda$ und $\overline{\mathcal{V}_\lambda}' := \mathcal{L}_{A^{-1}}^\sigma(\overline{\mathcal{V}_\lambda})$ ein zu $\overline{\mathcal{V}_\lambda}$ gleichwertiges Repräsentantensystem ist. Damit gilt, für $\overline{u} \in \mathcal{U}_\lambda$ und $0 \leq i \leq m$:

$$\begin{aligned}\left|\{v' \in \overline{\mathcal{V}_\lambda}' : \mathfrak{h}(\mathcal{R}_A^\sigma(\overline{u})v') = i\}\right| &= \left|\{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(\mathcal{R}_A^\sigma(\overline{u})\mathcal{L}_{A^{-1}}^\sigma(v)) = i\}\right| = \\ &= \left|\{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(\sigma(\overline{u}v)) = i\}\right| = \left|\{v \in \overline{\mathcal{V}_\lambda} : \mathfrak{h}(\overline{u}v) = i\}\right|.\end{aligned}$$

[13] vgl. Definition 3.1, diese lässt sich natürlich auf ganz $\mathcal{U}_\lambda$ ausdehnen.

Damit ist die Gleichheit der Höhenverteilung von $\overline{u}$ und $\mathcal{R}_A^\sigma(\overline{u})$ gezeigt. Dass $\mathcal{R}_A^\sigma(\overline{\mathcal{U}_\lambda})$ wieder ein gleichwertiges Vertretersystem ist, folgt in Analogie zum Beweis von Lemma 3.5 aus $\mathcal{R}_A^\sigma(r\overline{u}) = \sigma(r)\mathcal{R}_A^\sigma(\overline{u})$ für $\overline{u} \in \mathcal{U}_\lambda{}^*$ und $r \in R^\times$.

□

Satz 3.7. *Sei $n_0 < n$ und seien d und w wie üblich definiert. Sind $\Gamma, \Psi \in R^{k\times n_0}$ semilinear äquivalente Generatormatrizen zum Umriss $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1})$, dann gilt $\mathcal{E}(\Gamma) = \mathcal{E}(\Psi)$.*

Beweis. Da Γ und Ψ semilinear äquivalent sind, gibt es nach Korollar 2.24 $A \in \mathcal{A}^\lambda$ und $h = (r, (\pi, \sigma)) \in \mathcal{H}_{n_0}$ mit $r = (r_0, r_1, \ldots, r_{n_0-1})$, so dass $\Psi = A(h * \Gamma)$. Für $\overline{u} \in \overline{\mathcal{U}_\lambda}$ sei hier $\overline{u}'$ der Vertreter aus $\overline{\mathcal{U}_\lambda}$ zu $\mathcal{R}_{A^{-1}}^\sigma(\overline{u})$, etwa $\overline{u}' = s_{\overline{u}}\overline{\sigma(u)A^{-1}}$ mit $s_{\overline{u}} \in R^\times$ geeignet. Die j-te Spalte von Γ und Ψ sei mit γ_j beziehungsweise ψ_j bezeichnet. Aus Lemma 3.6 folgt, dass mit $\overline{u}$ auch $\overline{u}'$ ganz $\overline{\mathcal{U}_\lambda}$ durchläuft und dass $\overline{u}'$ dieselbe Höhenverteilung wie $\overline{u}$ besitzt. Daher gilt $\epsilon_{\overline{u}'} = \epsilon_{\overline{u}}$ und wir haben:

$$
\begin{aligned}
\mathcal{E}(\Psi) &= \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}'}(n-n_0, d - \sum_{j=0}^{n_0-1} w(\overline{u}'\psi_j)) = \\
&= \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}'}(n-n_0, d - \sum_{j=0}^{n_0-1} w(\overline{u}' A(\sigma(\gamma_{\pi^{-1}(j)})r_j^{-1}))) = \\
&= \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}}(n-n_0, d - \sum_{j=0}^{n_0-1} w(s_{\overline{u}}\overline{\sigma(u)A^{-1}} A(\sigma(\gamma_{\pi^{-1}(j)})r_j^{-1}))) \overset{s_{\overline{u}},\, r_j^{-1}\in R^\times}{=} \\
&= \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}}(n-n_0, d - \sum_{j=0}^{n_0-1} w(\sigma(u\gamma_{\pi^{-1}(j)}))) \overset{\mathfrak{h}(\sigma(x))=\mathfrak{h}(x)}{=} \\
&= \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}}(n-n_0, d - \sum_{j=0}^{n_0-1} w(u\gamma_{\pi^{-1}(j)})) = \\
&= \prod_{\overline{u}\in\overline{\mathcal{U}_\lambda}} \epsilon_{\overline{u}}(n-n_0, d - \sum_{j=0}^{n_0-1} w(u\gamma_j)) = \mathcal{E}(\Gamma).
\end{aligned}
$$

□

Satz 3.8. *Sind Γ und $\Psi \in R^{k\times n_0}$ semilinear äquivalente Generatormatrizen, so gibt es eine Bijektion $\phi : \overline{\mathcal{V}_\lambda} \to \overline{\mathcal{V}_\lambda}$, so dass für beliebiges $n' \le n$ und beliebige $\gamma_{n_0}, \ldots, \gamma_{n'-1} \in \overline{\mathcal{V}_\lambda}$ unter den Setzungen*

$$\Gamma' := \left(\ \Gamma \ \middle|\ \gamma_{n_0} \ \middle|\ \gamma_{n_0+1} \ \middle|\ \ldots \ \middle|\ \gamma_{n'-1}\ \right) \text{ und } \Psi' := \left(\ \Psi \ \middle|\ \phi(\gamma_{n_0}) \ \middle|\ \phi(\gamma_{n_0+1}) \ \middle|\ \ldots \ \middle|\ \phi(\gamma_{n'-1})\ \right)$$

gilt: $\mathcal{E}(\Gamma') = \mathcal{E}(\Psi')$.

Beweis. Sei Ψ dargestellt durch $\Psi = A((r,(\pi,\sigma)) * \Gamma)$ mit $r = (r_0, r_1, \ldots, r_{n_0-1})$. Für $\gamma \in \overline{\mathcal{V}_\lambda}$ sei $\phi(\gamma)$ der Vertreter aus $\overline{\mathcal{V}_\lambda}$ zu $\mathcal{L}_A^\sigma(\gamma)$. Nach Lemma 3.5 ist ϕ eine Bijektion. Für $n' \leq n$ definieren wir $r' \in (R^\times)^{n'}$, $\pi' \in S_{n'}$ durch

$$r' = (r_0, r_1, \ldots, r_{n_0-1}, 1, 1, \ldots, 1),\ \pi'(j) = \pi(j) \text{ für } j < n_0,\ \pi'(j) = j \text{ sonst.}$$

Für beliebige $\gamma_{n_0}, \ldots, \gamma_{n'-1} \in \overline{\mathcal{V}_\lambda}$ gilt dann $\Psi' = A((r',(\pi',\sigma)) * \Gamma')$. Ψ' und Γ' sind also semilinear isometrisch und nach Satz 3.7 gilt daher $\mathcal{E}(\Gamma') = \mathcal{E}(\Psi')$, wie behauptet. □

Da die von Algorithmus 3.3 im Teilbaum unterhalb von $x^{(i)}$ bzw. $y^{(i)}$ durchgeführten Schritte nur von den Bewertungen der spaltenweisen Erweiterungen von $\Gamma_{x^{(i)}}$ bzw. $\Gamma_{y^{(i)}}$ abhängen, ist nach Satz 3.8 klar, dass diese Teilbäume absolut gleichwertig sind, wenn $\Gamma_{x^{(i)}}$ und $\Gamma_{y^{(i)}}$ semilinear äquivalent sind.[14] Nun zur angekündigten heuristischen Methode, um diese Situationen zu erkennen:

3.7.2. Heuristische Erkennung semilinearer Isometrie

Um zuverlässig feststellen zu können, ob die aktuell betrachtete Generatormatrix semilinear isometrisch zu einer bereits vorher untersuchten ist, erscheint die aussichtsreichste Methode, ein Kanonisierungsverfahren einzusetzen, anstelle jeder Matrix den zugehörigen Repräsentanten (beziehungsweise einen daraus abgeleiteten Hashwert) zu speichern und im aktuellen Schritt nachzusehen, ob derselbe Repräsentant bereits zuvor aufgetreten ist. Allerdings ist die Kanonisierung von Generatormatrizen hinsichtlich semilinearer Isometrie schon über Körpern eine anspruchsvolle und rechenintensive Aufgabe [Feu09]. Um den heuristischen Algorithmus nicht unverhältnismäßig zu verlangsamen, müssten oft mehrere tausend Codes pro Sekunde kanonisiert werden, was schwer realisierbar erscheint. Wir wollen uns deshalb mit einer Heuristik begnügen, die nur ohnehin schon berechnete Werte verwendet und deshalb unter sehr geringen Geschwindigkeitsverlusten hinzugefügt werden kann. Sei die Multimenge $\mathcal{E}_\Gamma$ definiert durch

$$\mathcal{E}_\Gamma := \{\{\mathcal{E}(x_\Gamma + e_v) : v \in \overline{\mathcal{V}_\lambda}\}\}.$$

Nach Satz 3.8 gilt für semilinear isometrische Generatormatrizen Γ, Ψ: $\mathcal{E}_\Gamma = \mathcal{E}_\Psi$. Dies können wir so ausnutzen: Sei $\mathcal{P}$ die Menge aller Teilmultimengen von $\mathbb{R}_0^+$ und $h : \mathcal{P} \to \mathbb{R}_0^+$ eine Hashfunktion auf $\mathcal{P}$. Eine einfache Möglichkeit ist beispielsweise das elementweise Aufsummieren: Für eine Multimenge $\{\{\epsilon_0, \epsilon_1, \ldots, \epsilon_{r-1}\}\} \in \mathcal{P}$ ist dann

$$h(\{\{\epsilon_0, \epsilon_1, \ldots, \epsilon_{r-1}\}\}) := \sum_{i=0}^{r-1} \epsilon_r.$$

Nun befinde sich Algorithmus 3.3 auf Stufe i mit dem aktuellen Vektor $x^{(i)}$ und es sei $\Gamma_i := \Gamma_{x^{(i)}}$. Da beim Anlegen der Liste L_i ohnehin für jedes $v \in \overline{\mathcal{V}_\lambda}$ die Bewertung

[14] Lediglich Unterschiede in der Abarbeitungsreihenfolge bei gleich bewerteten Spaltenerweiterungen sind denkbar.

$\mathcal{E}(x^{(i)} + e_v)$ berechnet werden muss, liegen alle Werte von $\mathcal{E}_{\Gamma_i}$ vor, so dass auch $h(\mathcal{E}_{\Gamma_i})$ vergleichsweise einfach bestimmt werden kann. Ist $\mathcal{H}$ die Menge der im Laufe des Algorithmus bereits aufgetretenen Hashwerte, so wird nun geprüft, ob der Wert $h(\mathcal{E}_{\Gamma_i})$ bereits in $\mathcal{H}$ vorhanden ist. Falls ja, kehrt die Suche in der Annahme, dass Γ_i semilinear äquivalent zu einer bereits zuvor aufgetretenen Generatormatrix ist, direkt zum darunter liegenden Knoten zurück. Andernfalls wird $h(\mathcal{E}_{\Gamma_i})$ in $\mathcal{H}$ eingefügt und normal weiter verfahren. Es empfiehlt sich, die Hashfunktion h so zu wählen, dass sich ihr Wert bei Hinzufügen bzw. Entfernen von Nullelementen aus der Argumentmenge nicht ändert[15]. Dann ist das Verfahren direkt kompatibel mit der Entfernung von Variablen aus dem System nach Lemma 3.3, da Spalten mit Bewertung 0 auch bei der Berechnung des Hashwerts ignoriert werden können. Natürlich handelt es sich bei $\mathcal{E}_\Gamma = \mathcal{E}_\Psi$ nicht um ein *hinreichendes*, sondern nur um ein *notwendiges* Kriterium für die semilineare Äquivalenz von Γ und Ψ. Dies und - zumindest theoretisch - denkbare Hashkollisionen können zu fehlerhaften Äquivalenzannahmen und daher zu „unberechtigten“ Suchbaumreduktionen führen. Diese Fälle sind aber zum einen relativ unwahrscheinlich, zum anderen ist das heuristische Suchverfahren ohnehin nicht geeignet, die Nichtexistenz von Lösungen zu zeigen, da die Möglichkeit des Übersehens von Lösungen in der Natur der Sache liegt. Daher erscheint die Verwendung der vorgeschlagenen Heuristik äußerst ratsam.

[15]Dies ist zum Beispiel der Fall bei elementweiser Addition oder XOR-Verknüpfung.

Eingabe : **int** n, d: Blocklänge und Minimaldistanz
int t: Maximalwert von w auf R
int $\omega_0^{\overline{u}}, \omega_1^{\overline{u}}, \ldots, \omega_t^{\overline{u}}$: Gewichtsverteilung von $\overline{u}$
Ausgabe : Tabelleneinträge $\epsilon_{\overline{u}}[n'][d']$ für $0 \leq n' \leq n$, $0 \leq d' \leq d$

```
1  real ε_ū[0..n][0..d]; // globale Tabelle
2  int ν[0..t]; // repräsentiert aktuelle Gewichtsverteilung
3  procedure computeEpsilonTable(...)
4  {
5    ε_ū[0..n][0..d] ← 0.0; // initialisiere Tabelle mit 0
6    ν[0..t] ← 0; // initialisiere Gewichtsverteilung mit 0
7    recursiveComputation(0,0,0); // Aufruf der rekursiven Hauptfunktion
8    for int i from 0 to n do
9      for int j from d − 1 to 0 do
10       ε_ū[i][j] ← ε_ū[i][j] + ε_ū[i][j + 1]; // Aufsummieren
11     end for
12     for int j from d to 0 do
13       ε_ū[i][j] ← ε_ū[i][j]/ε_ū[i][0]; // Normierung
14     end for
15   end for
16 }
17 procedure recursiveComputation(int n', int d', int w)
18 {
19   int d̄' ← min{d', d};
20   ε_ū[n'][d̄'] ← ε_ū[n'][d̄'] + ∏_{i=0}^{t} C(ν[i]+ω_i^ū−1, ν[i]); // vgl. Kommentar im Text
21   if n' = n then
22     return;
23   end if
24   for int j from w to t do // w untere Schranke für nächstes Gewicht
25     if ω_j^u = 0 then
26       continue;
27     end if
28     ν[j] ← ν[j] + 1;
29     recursiveComputation(n' + 1,d' + j,j);
30   end for
31 }
```

Algorithmus 3.2: Vorberechnung der Werte von $\epsilon_{\overline{u}}$

Eingabe : **chainring** R: Grundring des zu konstruierenden Codes C
int $n, k, \lambda_0, \lambda_1, \ldots, \lambda_{k-1}, d$: Zielparameter für C
weightfunction w: Gewichtsfunktion
real $a \geq 1$: vom Benutzer festzulegender Parameter
vector $\xi \in \mathbb{N}^{\mathfrak{v}}$: vorgegebener Startvektor mit $\mathbb{1}^T\xi < n$
Ausgabe : **failed** oder **vector** $x \in \mathbb{N}^{\mathfrak{v}}$ mit $x > \xi$, $\mathbb{1}^T x = n$ und $\mathcal{M}^w x \geq d \cdot \mathbb{1}$

```
// erster Aufruf erfolgt mit backtrackAlg(ξ, 1^T ξ)
1  procedure backtrackAlg(vector x^(i), int i) : vector bzw. failed
2  {
3    if i = n then // Suche erfolgreich
4      return x^(i);
5    end if
6    list L_i ← ∅; // L_i speichert Paare aus Variable und Bewertung
7    foreach v ∈ V_λ do
8      pushBack(L_i, (v, E(x^(i) + e_v))); // Tupel anfügen
9    end foreach
10   sortDescending(L_i,2); // L_i absteigend nach 2. Komponente sortieren
11   (v*, β*) ← L_i.front(); // bestbewertetes Tupel auslesen
12   if β* = 0 then
13     return failed; // von hier aus keine Lösungen mehr möglich
14   end if
15   while L_i ≠ ∅ do
16     (v, β) ← L_i.front(); // vorderstes Tupel in (v, β) speichern...
17     L_i.popFront(); // ... und aus L_i nehmen
18     if a · β ≥ β* then
19       vector r ← backtrackAlg(x^(i) + e_v, i + 1);
20       if r ≠ failed then // Lösung ggf. weiterreichen
21         return r;
22       end if
23     end if
24   end while
25   return failed;
26 }
```

Algorithmus 3.3: Backtracking-Algorithmus

Kapitel 4.

Verallgemeinerung der Heuristik auf Kramer-Mesner-Systeme

Mitte der siebziger Jahre veröffentlichten Kramer und Mesner in [KM76] eine wegweisende Methode, um die bei der Suche nach t-Designs auftretenden Gleichungssysteme durch Vorschreiben einer Automorphismengruppe für die Designs deutlich zu reduzieren. Die Idee lässt sich auf viele andere kombinatorische Strukturen übertragen. Für lineare Codes über Körpern geschah das zum Beispiel in [Bra04] und [BKW05]. Wir wollen hier zunächst eine kurze Zusammenfassung für diesen Fall geben, anschließend eine Verallgemeinerung auf die Situation bei linearen Codes über Kettenringen vornehmen und schließlich Algorithmus 3.3 aus Kapitel 3 so modifizieren, dass er auf die dabei entstehenden Ungleichungssysteme angewendet werden kann.

4.1. Die Kramer-Mesner-Methode bei linearen Codes über Körpern

In diesem Abschnitt sei $R = \mathbb{F}_q$ ein Körper, $w = w_{\text{Ham}}$ und n, k, d seien die Werte für Blocklänge, Dimension und Minimaldistanz des angestrebten linearen Codes. Wir wollen das Konstruktionsproblem diesmal aus der geometrischen Perspektive betrachten und verweisen für die grundlegenden Definitionen und Zusammenhänge auf Abschnitt 2.4.1.

Auf den Elementen von $\mathrm{PG}(k-1,q)$ - den linearen Unterräumen U von $\mathbb{F}_q^k$ - operiert $\mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R)$ wie folgt:

$$(A,\sigma) * U := \{A\sigma(u) : u \in U\} \; \forall (A,\sigma) \in \mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R).$$

Man überzeugt sich leicht, dass dies tatsächlich eine wohldefinierte Gruppenoperation auf $\mathrm{PG}(k-1,q)$ ist, die außerdem die Dimension der Unterräume erhält. Insbesondere werden also Punkte in Punkte und Hyperebenen in Hyperebenen überführt. Sie lässt sich in der üblichen Weise zu einer Operation auf der Potenzmenge von $\mathrm{PG}(k-1,q)$ erweitern durch die Setzung $(A,\sigma) * T := \{(A,\sigma) * t : t \in T\}$ für $T \subset \mathrm{PG}(k-1,q)$.

Definition 4.1. Sei $\mathcal{S} \subset \mathcal{P}$ eine Multimenge von Punkten in $\mathrm{PG}(k-1,q)$. Die Menge

$$\mathrm{Aut}(\mathcal{S}) := \{(A,\sigma) \in \mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R) : (A,\sigma) * \mathcal{S} = \mathcal{S}\},$$

also der Mengenstabilisator von $\mathcal{S}$ unter der Operation von $\mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R)$, heißt die *Automorphismengruppe* von $\mathcal{S}$.

Es sei an dieser Stelle daran erinnert, dass die Suche nach einem linearen (n, q^k, d) − *Code* über $\mathbb{F}_q$ äquivalent ist zur Suche nach einem $(n, n-d)$-Arc $\mathfrak{t}$ in $\mathrm{PG}(k-1,q)$ (vgl. Lemma 2.11). Die Methode von Kramer und Mesner basiert nun auf der Idee, sich bei der Suche nach einem solchen Arc auf diejenigen Punktmultimengen $\mathcal{S} \subset \mathcal{P}$ zu beschränken, die bestimmten Symmetrien genügen, also auf solche, deren Automorphismengruppe eine vorgegebene Untergruppe $G \leq \mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R)$ enthält. Da ein solches $\mathcal{S}$ mit jedem Punkt aus $\mathrm{PG}(k-1,q)$ auch dessen gesamte Bahn[1] enthält, reduziert sich die Variablenanzahl des Systems auf die Zahl der Bahnen der Operation von G auf $\mathcal{P}$. In der ursprünglichen Sichtweise entspricht dies dem Zusammenfassen von Spalten von $\mathcal{M}^w$ im System (2.3) durch Aufaddieren, nämlich derjenigen, deren Labels aus $\overline{\mathcal{V}_\lambda}$ in derselben Bahn unter G liegen. Auch die Einträge in der letzten Zeile des Systems werden aufsummiert, so dass dort nun statt Einsen die jeweiligen Bahnlängen stehen. Die Reduktion der Variablenanzahl alleine wäre allerdings noch nichts Besonderes, schließlich ließe sich der Effekt auch durch willkürliche Einteilung der Spalten in Blöcke erreichen. Der entscheidende Vorteil bei Zusammenfassung zu Bahnen unter der Operation von G ist, dass auch die Zahl der Ungleichungen reduziert werden kann: Weil für beliebiges $\mathbf{p} \in \mathcal{P}$, $\mathbf{h} \in \mathcal{H}$ und $(A,\sigma) \in G$ die Beziehung

$$\mathbf{p} \text{ liegt auf } \mathbf{h} \Leftrightarrow (A,\sigma) * \mathbf{p} \text{ liegt auf } (A,\sigma) * \mathbf{h}$$

gilt, führt das Aufaddieren der Spalten nach Kramer und Mesner dazu, dass Zeilen zu Hyperebenen aus derselben Bahn identisch sind und daher jeweils nur ein einziger Vertreter im System belassen werden muss.

Die nach den Zeilen- und Spaltenreduktionen entstehende Matrix, die *Kramer-Mesner-Matrix* zur Gruppe G, wollen wir mit $\mathcal{M}_G^w$ bezeichnen. Ist $\omega = (\omega_0, \ldots, \omega_{s-1}) \in (\mathbb{N}^*)^s$ der Vektor der Bahnlängen von G auf $\mathcal{P}$, so erhält man insgesamt das Ungleichungssystem

$$\begin{aligned} \mathcal{M}_G^w y &\geq \begin{pmatrix} d \\ d \\ \vdots \\ d \end{pmatrix} \\ \omega^T y &= \quad n \end{aligned} \tag{4.1}$$

Bemerkung 4.1. Die Zahl der Bahnen von G auf den Punkten von $\mathrm{PG}(k-1,q)$ gleicht stets derjenigen auf den Hyperebenen[2], daher ist $\mathcal{M}_G^w$ für $R = \mathbb{F}_q$ quadratisch.

4.1.1. Berechnung der Bahnen von G

Ist $E = \{(A_1,\sigma_1), (A_2,\sigma_2), \ldots, (A_l,\sigma_l)\}$ ein Erzeugendensystem von G, so lässt sich die Bahn von G auf einem Punkt $\mathbf{p}(v_0) \in \mathcal{P}$, $v_0 \in \overline{\mathcal{V}_\lambda}$, mit Hilfe von Algorithmus 4.1

[1] mit entsprechender Vielfachheit

[2] Ein Beweis findet sich z. B. in [CK].

berechnen[3]. Durch sukzessives Anwenden auf die noch nicht in den bisher bestimmten Bahnen enthaltenen Punkte erhält man alle Bahnen von G auf $\mathcal{P}$. Die Operation von $\mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R)$ auf den Hyperebenen wird durch

$$\begin{aligned}(A,\sigma) * \mathbf{h}(\overline{u}) &= \{A\sigma(v) : v \in u^{\perp}\} = \{A\sigma(v) : v \in \mathbb{F}_q^k, uv = 0\} = \\ &= \{Av : v \in \mathbb{F}_q^k, u\sigma^{-1}(v) = 0\} = \{Av : v \in \mathbb{F}_q^k, \sigma(u)v = 0\} = \\ &= \{v \in \mathbb{F}_q^k : (\sigma(u)A^{-1})v = 0\} = \mathbf{h}(\overline{\sigma(u)A^{-1}})\end{aligned}$$

beschrieben. Die Bahn der Hyperebene $\mathbf{h}(\overline{u_0})$ für $\overline{u_0} \in \overline{\mathcal{U}_\lambda}$ erhält man also durch Anwenden von Algorithmus 4.1 mit $M := \mathcal{H}$, $m := \mathbf{h}(\overline{u_0})$, ${}_GM := {}_G\mathcal{H}$ und der Verknüpfung $(A,\sigma) * \mathbf{h}(\overline{u}) := \mathbf{h}(\overline{\sigma(u)A^{-1}})$ für $(A,\sigma) \in G$ und $\mathbf{h}(\overline{u}) \in \mathcal{H}$. Wir zeigen noch einen anderen Weg, der die Berechnung der Bahnen von G auf $\mathcal{H}$ auf die der Bahnen einer Gruppe G' auf $\mathcal{P}$ zurückführt. Folgende Definition gelte genauso für beliebige Kettenringe R:

Definition 4.2. Mit ϱ_V^l bezeichnen wir die Funktion, die jedem Element aus ${\mathcal{V}_\lambda}^*$ den Vertreter aus $\overline{\mathcal{V}_\lambda}$ unter der Operation von $R^\times$ per Multiplikation von links zuordnet. Entsprechend liefert ϱ_V^r den Vertreter bezüglich der Operation von $R^\times$ per Multiplikation von rechts.[4] Außerdem definieren wir ϱ_U^l als die Einschränkung von $\varphi_\lambda \circ \varrho_V^l \circ \varphi_\lambda^{-1}$ auf ${\mathcal{U}_\lambda}^*$. ϱ_U^l liefert daher zu jedem $\overline{u}$ in ${\mathcal{U}_\lambda}^*$ einen Vertreter aus $\overline{\mathcal{U}_\lambda}$ bezüglich der Linksmultiplikation mit Einheiten.

Bemerkung 4.2. Da R hier ein Körper ist, sind ϱ_V^l und ϱ_V^r natürlich gleich. Wir verwenden die beiden Symbole im Weiteren aber schon gemäß den Erfordernissen im nichtkommutativen Fall, da die Betrachtungen im Rest dieses Abschnitts ohnehin nur zur leichteren Orientierung in der Situation bei allgemeinen Kettenringen dienen.

$\mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R)$ operiert auf $\overline{\mathcal{V}_\lambda}$ vermöge

$$(A,\sigma) * v := \varrho_V^r(A\sigma(v))$$

und auf $\overline{\mathcal{U}_\lambda}$ mittels

$$(A,\sigma) * \overline{u} := \varrho_U^l(\overline{\sigma(u)A^{-1}}).$$

Die Mengen $\mathcal{P}$ und $\overline{\mathcal{V}_\lambda}$ bzw. $\mathcal{H}$ und $\overline{\mathcal{U}_\lambda}$ sind offensichtlich isomorph bezüglich der angeführten Gruppenoperationen, so dass wir im Folgenden nur noch die Bahnen von $\mathrm{GL}(k,q) \rtimes \mathrm{Aut}(R)$ auf $\overline{\mathcal{V}_\lambda}$ und $\overline{\mathcal{U}_\lambda}$ betrachten. Da R ein Körper ist, sind φ_λ und die Inverse φ_λ^{-1} gegeben durch

$$\varphi_\lambda : \overline{\mathcal{V}_\lambda} \to \overline{\mathcal{U}_\lambda}, v \mapsto \overline{v^T},\ \varphi_\lambda^{-1} : \overline{\mathcal{U}_\lambda} \to \overline{\mathcal{V}_\lambda}, \overline{u} \mapsto u^T.$$

Außerdem folgt aus der Definition von ϱ_U^l:

$$\varrho_U^l \circ \varphi_\lambda = \varphi_\lambda \circ \varrho_V^l.$$

[3]Hierzu muss $M := \mathcal{P}$, $m := \mathbf{p}(v_0)$ und ${}_GM := {}_G\mathcal{P}$ gesetzt werden mit der Verknüpfung $(A,\sigma) * \mathbf{p}(v) := \mathbf{p}(A\sigma(v))$ für $(A,\sigma) \in G$, $\mathbf{p}(v) \in \mathcal{P}$.

[4]Man beachte, dass $\overline{\mathcal{V}_\lambda}$ ein Vertretersystem von ${\mathcal{V}_\lambda}^*$ sowohl als R-Links- als auch als R-Rechtsmodul ist (vgl. Korollar 2.10), die Vertreter zu einem $v \in {\mathcal{V}_\lambda}^*$ können sich jedoch im nichtkommutativen Fall unterscheiden.

```
Eingabe : ${}_GM$: Gruppenoperation von $G$ auf $M$ (Verknüpfung $*$)
          $m \in M$: Element, zu dem die Bahn bestimmt werden soll
          set $E = \{g_1, g_2, \ldots, g_l\}$: Erzeugendensystem von $G$
Ausgabe : set $O$: Bahn von $m$ unter der Gruppenoperation ${}_GM$
 1 procedure orbitAlgorithm($m$) : set
 2 {
 3   set $O \leftarrow \{m\}$; // Menge der Bahnelemente
 4   set $N \leftarrow \{m\}$; // neu hinzugekommene Elemente
 5   while $N \neq \emptyset$ do
 6     $a \leftarrow$ pop($N$); // nehme Element aus $N$ und speichere es in $a$
 7     foreach $g \in E$ do
         // wende jedes $g \in E$ auf $a$ an
         if $g * a \notin O$ then
           // neues Element gefunden
           $O \leftarrow O \cup (g * a)$;
 8         $N \leftarrow N \cup (g * a)$;
 9       end if
10     end foreach
11   end while
12   return $O$;
13 }
```

Algorithmus 4.1: Bahnalgorithmus

Es gilt daher für $\overline{u} \in \overline{\mathcal{U}_\lambda}$:

$$\begin{aligned}(A,\sigma) * \overline{u} = \varrho_U^l(\overline{\sigma(u)A^{-1}}) = \varrho_U^l(\overline{((A^{-1})^T\sigma(u)^T)^T}) = \varrho_U^l(\varphi_\lambda((A^{-1})^T\sigma(u)^T)) = \\ = \varrho_U^l(\varphi_\lambda((A^{-1})^T\sigma(u^T))) = \varrho_U^l(\varphi_\lambda((A^{-1})^T\sigma(\varphi_\lambda^{-1}(\overline{u})))) = \\ = \varphi_\lambda(\varrho_V^l((A^{-1})^T\sigma(\varphi_\lambda^{-1}(\overline{u})))) = \varphi_\lambda(((A^{-1})^T,\sigma) * \varphi_\lambda^{-1}(\overline{u})).\end{aligned}$$

$\varphi_\lambda^{-1}(\overline{u})$ durchläuft hier ganz $\overline{\mathcal{V}_\lambda}$, wenn $\overline{u}$ alle Elemente von $\overline{\mathcal{U}_\lambda}$ durchläuft. Hat man also bereits einen Algorithmus, der zu dem Erzeugendensystem E von G die Bahnen auf $\overline{\mathcal{V}_\lambda}$ berechnet, so kann man diesen ebenso für die Berechnung der Bahnen auf $\overline{\mathcal{U}_\lambda}$ einsetzen: Man ersetzt zunächst jeden Erzeuger $e = (A,\sigma) \in E$ durch $e' := ((A^{-1})^T,\sigma)$ und berechnet die Bahnen der von den e' erzeugten Gruppe G' auf den Spaltenvektoren; anschließend überträgt man das Ergebnis durch Anwendung von φ_λ nach $\overline{\mathcal{U}_\lambda}$. Um das eventuell rechenintensive Invertieren der Matrizen bei den Erzeugern zu vermeiden, kann man stattdessen auch deren Inverse, die natürlich ebenfalls G generieren, verwenden. Das Inverse von e ist gegeben durch $e^{-1} = ((\sigma^{-1}(A))^{-1},\sigma^{-1})$ und man erhält

$$e^{-1} * \overline{u} = \varphi_\lambda((\sigma^{-1}(A)^T,\sigma^{-1}) * \varphi_\lambda^{-1}(\overline{u})).$$

Beispiel 4.1. Sei $R = \mathbb{F}_3$, $k = 3$ und $w = w_{\mathrm{Ham}}$. Dann ist[5]

$$\overline{\mathcal{V}_\lambda} = \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \right\}$$

und $\overline{\mathcal{U}_\lambda} = \overline{\mathcal{V}_\lambda}^T$. $\mathcal{M}^w$ ist der innere Bereich der folgenden Tabelle, die das Hammingge-wicht des Standardskalarprodukts zwischen den Elementen aus $\overline{\mathcal{U}_\lambda}$ und $\overline{\mathcal{V}_\lambda}$ auflistet:

$\overline{\mathcal{U}_\lambda}/\overline{\mathcal{V}_\lambda}$	0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 2 2 2 1 0 1 2 0 1 2 0 1 2 0 1 2
0 0 1	1 0 1 1 0 1 1 0 1 1 0 1 1
0 1 0	0 1 1 1 0 0 0 1 1 1 1 1 1
0 1 1	1 1 1 0 0 1 1 1 1 0 1 0 1
0 1 2	1 1 0 1 0 1 1 1 0 1 1 1 0
1 0 0	0 0 0 0 1 1 1 1 1 1 1 1 1
1 0 1	1 0 1 1 1 1 0 1 1 0 1 1 0
1 0 2	1 0 1 1 1 0 1 1 0 1 1 0 1
1 1 0	0 1 1 1 1 1 1 1 1 1 0 0 0
1 1 1	1 1 1 0 1 1 0 1 0 1 0 1 1
1 1 2	1 1 0 1 1 0 1 1 1 0 0 1 1
1 2 0	0 1 1 1 1 1 1 0 0 0 1 1 1
1 2 1	1 1 0 1 1 1 0 0 1 1 1 0 1
1 2 2	1 1 1 0 1 0 1 0 1 1 1 1 0

Sei nun G die von (A, id) erzeugte Untergruppe, wobei die Matrix A gegeben ist durch

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 1 \\ 1 & 2 & 1 \end{pmatrix}.$$

[5]Der einfacheren Lesbarkeit halber verzichten wir hier auf die Querstriche bei der Darstellung der Elemente von $\mathbb{F}_3$.

Die Bahnen von G auf den Punkten beziehungsweise Hyperebenen sind:

$$G\backslash\backslash\mathcal{P} = \left\{\left\{\left\langle\begin{pmatrix}0\\0\\1\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}0\\1\\1\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\0\\0\end{pmatrix}\right\rangle\right\},\left\{\left\langle\begin{pmatrix}0\\1\\0\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\0\\2\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\2\\2\end{pmatrix}\right\rangle\right\},\right.$$
$$\left\{\left\langle\begin{pmatrix}0\\1\\2\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\0\\1\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\1\\1\end{pmatrix}\right\rangle\right\},\left\{\left\langle\begin{pmatrix}1\\1\\0\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\2\\0\end{pmatrix}\right\rangle,\left\langle\begin{pmatrix}1\\2\\1\end{pmatrix}\right\rangle\right\},$$
$$\left.\left\{\left\langle\begin{pmatrix}1\\1\\2\end{pmatrix}\right\rangle\right\}\right\},$$

$$G\backslash\backslash\mathcal{H} = \left\{\left\{(001)^{\perp},(110)^{\perp},(121)^{\perp}\right\},\left\{(010)^{\perp},(012)^{\perp},(100)^{\perp}\right\},\left\{(101)^{\perp}\right\},\right.$$
$$\left.\left\{(011)^{\perp},(112)^{\perp},(120)^{\perp}\right\},\left\{(102)^{\perp},(111)^{\perp},(122)^{\perp}\right\}\right\}.$$

Sei $n = 7$ und $d = 4$. Das zu lösende Ungleichungssystem nach Vorschreiben von G als Automorphismenuntergruppe lautet dann

$$\begin{pmatrix}2&2&3&1&1\\1&2&2&3&1\\2&3&2&2&0\\3&0&3&3&0\\3&2&1&2&1\end{pmatrix}\cdot x \geq \begin{pmatrix}4\\4\\4\\4\\4\end{pmatrix}$$

$$\begin{pmatrix}3&3&3&3&1\end{pmatrix}\cdot x = 7$$

Eine Lösung ist $x^T = \begin{pmatrix}0&0&1&1&1\end{pmatrix}$ und eine mögliche Generatormatrix Γ für einen korrespondierenden linearen Code

$$\Gamma = \left(\begin{array}{ccc|ccc|c}0&1&1&1&1&1&1\\1&0&1&1&2&2&1\\2&1&1&0&0&1&2\end{array}\right).$$

4.2. Kramer-Mesner für lineare Codes über Kettenringen

Ab jetzt sei R wieder ein allgemeiner endlicher Kettenring, w eine beliebige Gewichtsfunktion und λ ein Umriss. Die oben vorstellte Methode von Kramer und Mesner soll nun von Codes über endlichen Körpern auf solche über endlichen Kettenringen verallgemeinert werden. Eine direkte Übersetzung in der Sprache der Geometrie ist an

dieser Stelle nicht möglich, da zum einen auch nichtfreie Codes betrachtet werden und außerdem nichtfette Vektoren in der Generatormatrix bzw. deren Äquivalente in der Informationsvektormenge zulässig sein sollen. Dagegen sind Punkte und Hyperebenen in der projektiven Hjelmslev-Geometrie per Definition stets freie (Rechts-)Untermoduln von R_R^k. Das Problem lässt sich aber relativ einfach lösen: Zwar kann man, anders als im Falle der Körper, die Elemente aus $\overline{\mathcal{V}_\lambda}$ bzw. $\overline{\mathcal{U}_\lambda}$ nicht mehr eins zu eins mit den Punkten und Hyperebenen identifizieren, es spricht jedoch nichts dagegen, die geometrische Sichtweise aufzugeben und stattdessen direkt die Operationen von $\mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$ auf den Mengen $\overline{\mathcal{V}_\lambda}$ und $\overline{\mathcal{U}_\lambda}$ zu betrachten. Zu $(A,\sigma) \in \mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$, $v \in \overline{\mathcal{V}_\lambda}$ und $\overline{u} \in \overline{\mathcal{U}_\lambda}$ sind diese Operationen gegeben durch

$$(A,\sigma) * v := \varrho_V^r(A\sigma(v)),\ (A,\sigma) * \overline{u} := \varrho_U^l(\overline{\sigma(u)A^{-1}}).$$

und lassen sich durch elementweise Anwendung wieder problemlos auf Teilmengen von $\overline{\mathcal{V}_\lambda}$ und $\overline{\mathcal{U}_\lambda}$ übertragen. Die Wohldefiniertheit der zweiten Operation zeigt das folgende Lemma:

Lemma 4.1. *Sei $A = (a_{ij}) \in \mathcal{A}^\lambda$ und $\overline{u}$ in $\mathcal{U}_\lambda$. Dann ist der Wert von $\overline{\sigma(u)A}$ unabhängig von der Wahl des Repräsentanten u.*

Beweis. Seien $x = (x_0, x_1, \ldots, x_{k-1})$ und $y = (y_0, y_1, \ldots, y_{k-1}) \in R^k$ mit $\overline{x} = \overline{y}$, also $y_i = x_i + r_i\theta^{\lambda_i}$ für geeignete $r_i \in R$. Dann gilt:

$$\begin{aligned}(\sigma(y)A)_j &= \sum_{i=0}^{k-1} \sigma(y_i)a_{ij} = \sum_{i=0}^{k-1}(\sigma(x_i + r_i\theta^{\lambda_i}))a_{ij} = \\ &= \sum_{i=0}^{j}(\sigma(x_i)a_{ij} + \underbrace{\sigma(r_i\theta^{\lambda_i})a_{ij}}_{\equiv 0,\ \text{da } \lambda_i \geq \lambda_j}) + \sum_{i=j+1}^{k-1}(\sigma(x_i)a_{ij} + \underbrace{\sigma(r_i\theta^{\lambda_i})a_{ij}}_{\equiv 0,\ \text{da } a_{ij} \in \mathrm{Rad}(R)^{\lambda_j - \lambda_i}}) \equiv \\ &\equiv (\sigma(x)A)_j \mod \mathrm{Rad}(R)^{\lambda_j}.\end{aligned}$$

Daraus folgt $\overline{\sigma(y)A} = \overline{\sigma(x)A}$, also die Behauptung.

□

Zu jeder Teilmultimenge $\mathcal{S}$ von $\overline{\mathcal{V}_\lambda}$ erhält man einen zugehörigen R-linearen Code, indem man jeden Vektor aus $\mathcal{S}$ gemäß seiner Vielfachheit in die Generatormatrix einträgt. Um R-lineare Codes mit der Kramer-Mesner-Methode zu konstruieren, gibt man sich nun eine Untergruppe G von $\mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$ vor und beschränkt sich auf solche Multimengen $\mathcal{S} \subset \overline{\mathcal{V}_\lambda}$, die invariant unter G sind, d. h. $G(\mathcal{S}) = \mathcal{S}$ erfüllen. Das ist äquivalent dazu, dass mit jedem $v \in \overline{\mathcal{V}_\lambda}$ gleich die gesamte Bahn von v unter G in $\mathcal{S}$ liegt und bedeutet wie im Abschnitt zuvor, dass in $\mathcal{M}^w$ die Spalten zu Vektoren aus derselben Bahn per Summation zusammengefasst werden. Da die Höhe der Elemente aus R unter jedem Ringautomorphismus $\sigma \in \mathrm{Aut}(R)$ unverändert bleibt, ergibt eine einfache Rechnung:

$$\mathfrak{h}(((A,\sigma) * \overline{u})((A,\sigma) * v)) = \mathfrak{h}(\overline{u}v). \tag{4.2}$$

Da w außerdem konstant auf den Elementen gleicher Höhe ist, folgt unmittelbar auch

$$w(((A,\sigma) * \overline{u})((A,\sigma) * v)) = w(\overline{u}v).$$

Deshalb reduziert sich wieder nicht nur die Variablenanzahl des Ungleichungssystems, sondern auch die Zeilenanzahl, da Zeilen zu Informationsvektoren aus derselben Bahn unter G nach der Spaltenreduktion identisch sind. Die so reduzierte Matrix nennen wir wie oben die Kramer-Mesner-Matrix zu G und notieren sie mit $\mathcal{M}_G^w$.

Bemerkung 4.3. Die beim Aufbau der Gruppendatenbank (vgl. Kapitel 5) gewonnenen experimentellen Daten lassen vermuten, dass es sich auch über allgemeinen endlichen Kettenringen bei $\mathcal{M}_G^w$ stets um eine quadratische Matrix handelt. Allerdings erscheint dies deutlich schwerer zu zeigen als im Körperfall, beispielsweise ist nicht unmittelbar klar, wie sich der Beweis aus [CK] übertragen lässt. Da dieser Aspekt aber für den weiteren Verlauf der Arbeit keine Rolle spielt, wollen wir ihn hier nicht näher untersuchen.

Die Bahnen von G auf $\overline{\mathcal{V}_\lambda}$ und $\overline{\mathcal{U}_\lambda}$ können mit Algorithmus 4.1 berechnet werden. Wie zuvor besteht aber auch hier die Möglichkeit, die Berechnung der Bahnen auf den Informationsvektoren auf die der Bahnen auf den Spaltenvektoren zurückzuführen. Dies spart zum einen Implementierungsaufwand, zum anderen auch Laufzeit, da die häufige Anwendung von φ_λ bzw. φ_λ^{-1} bei der Auswertung von ϱ_U^l entfällt. Ein Verfahren hierzu soll nun hergeleitet werden:

Lemma 4.2. *Seien $y, z \in R$. Es gelte $\mathfrak{h}(y) \leq \mathfrak{h}(z)$. Dann gibt es stets $x_l \in R$ mit $x_l y = z$ und ebenso existiert $x_r \in R$ mit $y x_r = z$. x_l und x_r sind dabei nur modulo $Rad(R)^{m-\mathfrak{h}(y)}$ festgelegt.*

Beweis. Wir beschränken uns auf die Behauptung für x_l, für x_r verläuft die Argumentation völlig analog. Die Abbildung $\mu_y : R \to R, x \mapsto xy$ ist ein Endomorphismus von R als R-Linksmodul und $\ker(\mu_y) = \mathrm{Rad}(R)^{m-\mathfrak{h}(y)}$. Das Bild von μ_y ist, als R-Linksmodul, isomorph zu $R/\ker(\mu_y)$, also ein Linksideal von R der Ordnung $q^{m-\mathfrak{h}(y)}$. Aus Satz 1.3 folgt dann $\mu_y(R) = \mathrm{Rad}(R)^{\mathfrak{h}(y)}$ und wegen $\mathfrak{h}(y) \leq \mathfrak{h}(z)$ ergibt sich $z \in \mu_y(R)$, insgesamt also die Behauptung.

□

Definition 4.3. Sei $\lambda = (\lambda_0, \ldots, \lambda_{k-1})$. Zu $A = (a_{ij}) \in \mathcal{A}^\lambda$ und $\sigma \in \mathrm{Aut}(R)$ wählen wir eine Matrix $B_{A,\sigma} = (b_{ij}) \in R^{k\times k}$ mit der Eigenschaft $\sigma(\theta^{m-\lambda_i})b_{ji} = a_{ij}\theta^{m-\lambda_j}$ für alle i, j mit $0 \leq i, j \leq k-1$. Da $a_{ij} \in \mathrm{Rad}(R)^{\lambda_j-\lambda_i}$ für $i > j$, folgt mit Lemma 4.2, dass eine solche Wahl der b_{ji} möglich und nur modulo $\mathrm{Rad}(R)^{\lambda_i}$ eindeutig ist.

Lemma 4.3. *Zu $A \in \mathcal{A}^\lambda$ und $\sigma \in \mathrm{Aut}(R)$ sei $B_{A,\sigma} = (b_{ij})$ wie in Definition 4.3 konstruiert. Ferner notiere $\bullet$ die duale Multiplikation in R, d. h. $ab = b \bullet a, \forall a, b \in R$. Dann gilt, für $\overline{u} \in \overline{\mathcal{U}_\lambda}$:*

$$\overline{\sigma(u)A} = \varphi_\lambda(B_{A,\sigma} \bullet \sigma(\varphi_\lambda^{-1}(\overline{u}))).$$

Beweis. Die j-te Komponente des Argumentes von φ_λ auf der rechten Seite ist

$$(B_{A,\sigma} \bullet \sigma(\varphi_\lambda^{-1}(\overline{u})))_j = \sum_{i=0}^{k-1} b_{ji} \bullet (\sigma(u_i \theta^{m-\lambda_i})) = \sum_{i=0}^{k-1} \sigma(u_i)(\sigma(\theta^{m-\lambda_i}) b_{ji}) =$$
$$\overset{\text{Def. von } b_{ji}}{=} \sum_{i=0}^{k-1} \sigma(u_i) a_{ij} \theta^{m-\lambda_j}.$$

Wendet man auf dieses Ergebnis noch φ_λ an („Division durch $\theta^{m-\lambda_j}$“), erhält man gerade die j-te Komponente der linken Seite der Gleichung.

□

Korollar 4.4. *Für $(A, \sigma) \in \mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$ und $\overline{u} \in \overline{\mathcal{U}_\lambda}$ gilt:*

$$(A, \sigma) * \overline{u} = \varphi_\lambda(\varrho_V^l(B_{A^{-1},\sigma} \bullet \sigma(\varphi_\lambda^{-1}(\overline{u})))).$$

Beweis.

$$(A, \sigma) * \overline{u} = \varrho_U^l(\overline{\sigma(u) A^{-1}}) \overset{\text{Satz 4.3}}{=} \varrho_U^l(\varphi_\lambda(B_{A^{-1},\sigma} \bullet \sigma(\varphi_\lambda^{-1}(\overline{u})))) \overset{\varrho_U^l \circ \varphi_\lambda = \varphi_\lambda \circ \varrho_V^l}{=}$$
$$= \varphi_\lambda(\varrho_V^l(B_{A^{-1},\sigma} \bullet \sigma(\varphi_\lambda^{-1}(\overline{u})))).$$

□

Wie schon im einfacheren Fall der endlichen Körper kann also für die Berechnung der Bahnen auf den Informationsvektoren grundsätzlich derselbe Algorithmus verwendet werden wie für die Bestimmung der Bahnen auf den Spaltenvektoren. Es müssen lediglich die Erzeugerelemente gemäß Korollar 4.2 umgerechnet ($(A, \sigma) \to (B_{A^{-1},\sigma}, \sigma)$) und die von ihnen auf $\overline{\mathcal{V}_\lambda}$ bezüglich der dualen Ringmultiplikation generierten Bahnen anschließend durch Anwendung von φ_λ nach $\overline{\mathcal{U}_\lambda}$ übertragen werden. Auch hier bietet es sich an, statt der ursprünglichen Erzeuger ihre Inversen zu verwenden, um ein Invertieren der Matrix A zu vermeiden.

Bemerkung 4.4. Möchte man Kramer-Mesner-Matrizen für verschiedene Gewichtsfunktionen bei sonst gleichen Parametern berechnen, empfiehlt es sich, $\mathcal{M}_G^\chi$ zunächst bezüglich der Funktion $\chi : R \to \{X_0, X_1, \ldots, X_m\}$, $r \mapsto X_{m-\mathfrak{h}(r)}$, zu ermitteln. χ heißt das *symmetrisierte Gewicht* auf R. Es handelt sich zwar trotz des Namens offensichtlich nicht um eine Gewichtsfunktion im Sinne von Definition 2.5, aber Gleichung (4.2) garantiert, dass dennoch dieselben Zeilenreduktionen beim Anwenden des Kramer-Mesner-Verfahrens zulässig sind. Wir nennen $\mathcal{M}_G^\chi$ die *symmetrisierte* Kramer-Mesner-Matrix. Für jede richtige Gewichtsfunktion w lässt sich $\mathcal{M}_G^w$ dann sehr leicht bestimmen, indem man den Eintrag X_i in $\mathcal{M}_G^\chi$ überall durch $w(\theta^{m-i})$ ersetzt.

Beispiel 4.2. Sei a ein primitives Element in $\mathbb{F}_4$, σ der Frobeniusautomorphismus auf $\mathbb{F}_4$ und $R = \mathbb{F}_4[X, \sigma]/(X^2)$. Nach Satz 1.15 ist R dann ein Kettenring mit $p = r = m = 2$,

$\alpha = a$ und $\theta = X$.[6] Sei $\lambda = (2,2)$ und $A \in \mathcal{A}^\lambda$ mit

$$A = \begin{pmatrix} aX & aX+a+1 \\ (a+1)X+1 & (a+1)X+a+1 \end{pmatrix}.$$

Man rechnet nach, dass durch die Festlegungen

$$a \mapsto aX + (a+1),\ X \mapsto (a+1)X$$

ein Automorphismus ψ auf R induziert wird, der der im Anhang A gegebenen Verknüpfungstafel genügt.

Sei G die von (A,ψ) erzeugte Untergruppe in $\mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$. Wir wollen zuerst die Bahnen der Operation von G auf der Menge der Spaltenrepräsentanten $\overline{\mathcal{V}_\lambda}$ bestimmen. Die Bahn des Elementes $v \in \overline{\mathcal{V}_\lambda}$ erhält man durch Anwenden der Potenzen von (A,ψ) auf v und anschließendes Normieren mit ϱ_V^r. Für das Element $(1,0)^T$ ergibt sich beispielsweise die Abfolge

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} aX \\ 1 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} 1 \\ aX+1 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} 1 \\ X+a+1 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} 1 \\ aX \end{pmatrix} \overset{(A,\psi)}{\to}$$
$$\overset{(A,\psi)}{\to} \begin{pmatrix} (a+1)X \\ 1 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} 1 \\ a+1 \end{pmatrix} \overset{(A,\psi)}{\to} \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Insgesamt erhält man als Bahnenmenge auf den Spaltenvertretern:

$$\begin{aligned} G\backslash\backslash\overline{\mathcal{V}_\lambda} = \Bigg\{ & \Bigg\{ \binom{1}{0}, \binom{aX}{1}, \binom{1}{aX+1}, \binom{1}{X+a+1}, \binom{1}{aX}, \binom{(a+1)X}{1}, \binom{1}{1}, \binom{1}{a+1} \Bigg\}, \\ & \Bigg\{ \binom{1}{a}, \binom{1}{X+a}, \binom{1}{(a+1)X+a}, \binom{1}{aX+a} \Bigg\}, \\ & \Bigg\{ \binom{1}{X}, \binom{0}{1}, \binom{1}{(a+1)X+1}, \binom{1}{(a+1)X+a+1}, \binom{1}{(a+1)X}, \binom{X}{1}, \binom{1}{X+1}, \\ & \quad \binom{1}{aX+a+1} \Bigg\}, \\ & \Bigg\{ \binom{X}{0}, \binom{0}{X}, \binom{X}{X}, \binom{X}{(a+1)X} \Bigg\}, \\ & \Bigg\{ \binom{X}{aX} \Bigg\} \Bigg\} \end{aligned}$$

Für die Bahnen auf den Informationsvektoren bestimmen wir zunächst

$$(A,\psi)^{-1} = ((\psi^{-1}(A))^{-1}, \psi^{-1}) = \left(\begin{pmatrix} aX & a \\ X+1 & (a+1)X+a \end{pmatrix}^{-1}, \psi^{-1} \right).$$

[6] Zur leichteren Nachvollziehbarkeit des Beispiels sind in Anhang A die Verknüpfungstafeln von $R = \mathbb{F}_4[X,\sigma]/(X^2)$ aufgeführt. Außerdem verzichten wir der besseren Lesbarkeit halber auf Querstriche über den Elementen von R bzw. $\overline{\mathcal{U}_\lambda}$. Mit X als Element in R ist also beispielsweise die Nebenklasse $\overline{X} = X + (X^2)$ gemeint, während $(X,0)$ als Element von $\overline{\mathcal{U}_\lambda}$ genau genommen für den Vektor $(\overline{X} + \{\overline{0}\}, \overline{0} + \{\overline{0}\})$ steht.

$B_{\psi^{-1}(A),\psi^{-1}}$ erhält man in diesem Beispiel durch einfache Transposition, da $\lambda_0 = \lambda_1 = m$ und somit $\theta^{m-\lambda_i} = 1$ für $i \in \{0, 1\}$ gilt:

$$B_{\psi^{-1}(A),\psi^{-1}} = \psi^{-1}(A)^T = \begin{pmatrix} aX & X+1 \\ a & (a+1)X+a \end{pmatrix}.$$

Nach Satz 4.3 gilt nun, für $u \in \mathcal{U}_\lambda$:[7]

$$\psi^{-1}(u)\psi^{-1}(A) = \left(B_{\psi^{-1}(A),\psi^{-1}} \bullet \psi^{-1}(u^T)\right)^T.$$

Für $u = (X, 1)$ ergibt das beispielsweise

$$\begin{aligned}\left(B_{\psi^{-1}(A),\psi^{-1}} \bullet \psi^{-1}((X,1)^T)\right)^T &= \left(B_{\psi^{-1}(A),\psi^{-1}} \bullet ((a+1)X, 1)^T\right)^T = \\ &= \left(\begin{pmatrix} aX & X+1 \\ a & (a+1)X+a \end{pmatrix} \bullet \begin{pmatrix} (a+1)X \\ 1 \end{pmatrix}\right)^T = \\ &= \begin{pmatrix} X+1 \\ X+a \end{pmatrix}^T = (X+1, X+a)\end{aligned}$$

und anschließende Normierung durch Linksmultiplikation mit $(X + 1)$ liefert dann den zugehörigen Repräsentanten $(1, aX + a)$. Die vollständige Bahnenmenge auf den Repräsentanten der Informationsvektoren ist

$$\begin{aligned} G\backslash\backslash\overline{\mathcal{U}_\lambda} = \Big\{ &\Big\{ (1,0), (X,1), (1,aX+a), (1,X+1), (1,X), ((a+1)X,1), (1,(a+1)X+a), \\ &\quad (1,(a+1)X+1) \Big\}, \\ &\Big\{ (1,1), (1,(a+1)X), (aX,1), (1,X+a), (1,aX+1), (1,aX), (0,1), (1,a) \Big\}, \\ &\Big\{ (1,a+1), (1,aX+a+1), (1,(a+1)X+a+1), (1,X+a+1) \Big\}, \\ &\Big\{ (X,0), (0,X), (X,(a+1)X), (X,X) \Big\}, \\ &\Big\{ (X,aX) \Big\} \Big\} \end{aligned}$$

Für die Aufstellung der Kramer-Mesner-Matrix bestimmen wir die Perioden der inneren Produkte zwischen den Bahnrepräsentanten der Informationsvektoren und den Spaltenvektoren und erhalten die in Anhang A gegebene Matrix $\mathcal{M}^\chi$. Nach bahnweisem

[7] Denn die Anwendung von $\varphi_\lambda : \mathcal{U}_\lambda \to \mathcal{V}_\lambda$ entspricht hier lediglich einer Transposition und die Projektion auf $\mathcal{U}_\lambda$ der Identität.

Zusammenaddieren der Spalten ergibt sich folgende Periodenverteilung:

	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ a \end{pmatrix}$	$\begin{pmatrix} 1 \\ X \end{pmatrix}$	$\begin{pmatrix} X \\ 0 \end{pmatrix}$	$\begin{pmatrix} X \\ aX \end{pmatrix}$
$(1,0)$	$6X_2+2X_1$	$4X_2$	$6X_2+X_1+X_0$	$3X_1+X_0$	X_1
$(1,1)$	$6X_2+X_1+X_0$	$4X_2$	$6X_2+2X_1$	$3X_1+X_0$	X_1
$(1,a+1)$	$8X_2$	$3X_1+X_0$	$8X_2$	$4X_1$	X_0
$(X,0)$	$6X_1+2X_0$	$4X_1$	$6X_1+2X_0$	$4X_0$	X_0
(X,aX)	$8X_1$	$4X_0$	$8X_1$	$4X_0$	X_0
Bahnlänge	8	4	8	4	1

Unter Verwendung des homogenen Gewichts ($X_0 \mapsto 0$, $X_1 \mapsto 4$, $X_2 \mapsto 3$) ergibt sich die Matrix $\mathcal{M}_G^{w_{\text{hom}}}$ als der innere Bereich der folgenden Tabelle:

	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ a \end{pmatrix}$	$\begin{pmatrix} 1 \\ X \end{pmatrix}$	$\begin{pmatrix} X \\ 0 \end{pmatrix}$	$\begin{pmatrix} X \\ aX \end{pmatrix}$
$(1,0)$	26	12	22	12	4
$(1,1)$	22	12	26	12	4
$(1,a+1)$	24	12	24	16	0
$(X,0)$	24	16	24	0	0
(X,aX)	32	0	32	0	0
Bahnlänge	8	4	8	4	1

Man erkennt, dass bei Aufaddieren der ersten drei Spalten die Zeilensumme überall 60 oder 64 beträgt. Das bedeutet, dass eine Generatormatrix, die alle Vektoren der ersten drei Bahnen genau einmal enthält, einen R-linearen Code *two-weight-Code*[8] mit Länge $8+4+8=20$ und den beiden Gewichten 60 und 64 erzeugt. Durch Anwenden der Grayabbildung erhält man hieraus einen optimalen linearen $(80, 4^4, 60)$-Code über $\mathbb{F}_4$.

4.3. Anpassung der Heuristik an Kramer-Mesner-Systeme

Nun wollen wir die in Kapitel 3 eingeführte Suchheuristik auf die bei Anwendung der Kramer-Mesner-Methode entstehenden Ungleichungssysteme anpassen. Diese fallen in die Klasse der etwas allgemeineren Systeme vom Typ

$$\begin{aligned} a := \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{pmatrix} \leq \mathcal{T}x \leq \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{k-1} \end{pmatrix} =: b \\ \omega^T x = \quad n \end{aligned} \tag{4.3}$$

[8]Ein Code heißt *two-weight-Code*, wenn die in ihm enthaltenen Worte - vom Nullwort abgesehen - nur zwei verschiedene Gewichte besitzen.

mit $\mathcal{T} \in \mathbb{Z}^{k\times l}$, $a_i \leq b_i \in \mathbb{Z}$, $\omega \in (\mathbb{N}^*)^l$, $n \in \mathbb{N}^*$ und $x \in \mathbb{N}^l$, mit denen wir uns in der Folge beschäftigen wollen. Bei den Kramer-Mesner-Systemen für lineare Codes über Kettenringen können die Einträge von b auf ∞ (bzw. entsprechend große Zahlen gesetzt werden), während in allen Komponenten von a die angestrebte Minimaldistanz steht. Die Werte von $\mathcal{T}$ sind dabei allesamt nichtnegativ.

4.3.1. Bewertungsfunktion bei Kramer-Mesner-Systemen

Im ersten Schritt soll die Bewertungsfunktion $\mathcal{E}$ verallgemeinert werden. Dazu nehmen wir eine Anpassung von Definition 3.2 vor:

Definition 4.4. Für das Ungleichungssystem (4.3), $i \in \{0, 1, \ldots, k-1\}$ und $x \in \mathbb{N}^l$ sei:

(i) $\mathscr{N}(x) := \{x' \geq x : x' \in \mathbb{N}^l, \omega^T x' = n\}$ die Menge der *zulässigen Nachfolger* von x.

(ii) $\mathscr{S}_i(x) := \{x' \in \mathscr{N}(x) : a_i \leq \mathcal{T}_{i*}x' \leq b_i\}$, die *Lösungsmenge zu x* für Zeile i. Für $I \subset \{0, 1, \ldots, k-1\}$ sei $\mathscr{S}_I(x)$ definiert durch

$$\mathscr{S}_I(x) := \bigcap_{i\in I} \mathscr{S}_i$$

und $\mathscr{S}(x) := \mathscr{S}_{\{0,1,\ldots,k-1\}}(x)$.

(iii) $\delta_i(x) := \frac{|\mathscr{S}_i(x)|}{|\mathscr{N}(x)|}$ die *Lösungsdichte zu x* für Zeile i und $\delta(x) := \frac{|\mathscr{S}(x)|}{|\mathscr{N}(x)|}$ einfach nur die *Lösungsdichte zu x*.[9]

Wie in Abschnitt 3.2 wollen wir auch hier annehmen, dass zwischen den Erfülltheitsbedingungen der einzelnen Ungleichungen stochastische Unabhängigkeit herrscht und setzen dementsprechend wieder

$$\mathscr{E}(x) := \prod_{i=0}^{k-1} \delta_i(x) \approx \delta(x).$$

Der Wert $\delta_i(x)$ hängt lediglich von i sowie $\omega^T x$ und $\mathcal{T}_{i*}x$ ab, denn

$$\delta_i(x) = \frac{\left|\{x' \in \mathbb{N}^l : \omega^T x' = n - \omega^T x,\ a_i - \mathcal{T}_{i*}x \leq \mathcal{T}_{i*}x' \leq b_i - \mathcal{T}_{i*}x\}\right|}{|\{x' \in \mathbb{N}^l : \omega^T x' = n - \omega^T x\}|}.$$

Dies erkennt man durch die Feststellung, dass die Abbildung $x' \mapsto x + x'$ jeweils eine Bijektion zwischen den Mengen in Zähler und Nenner der rechten Seite und $\mathscr{S}_i(x)$ bzw. $\mathscr{N}(x)$ ist.

Erneut in Analogie zu Kapitel 3.2 definieren wir[10]

[9] vgl. Bemerkung 4.5

[10] vgl. Bemerkung 4.5

$$\epsilon_i : \mathbb{N} \times \mathbb{Z} \to \mathbb{R}_0^+, (n', s) \mapsto \frac{\left|\{x' \in \mathbb{N}^l : \omega^T x' = n',\ a_i - s \leq \mathcal{T}_{i*} x' \leq b_i - s\}\right|}{|\{x' \in \mathbb{N}^l : \omega^T x' = n'\}|}$$

und es gilt damit:

$$\delta_i(x) = \epsilon_i(n - \omega^T x, \mathcal{T}_{i*} x).$$

Bemerkung 4.5. Sollten bei der Auswertung von $\delta_i(x)$, $\delta(x)$ oder $\epsilon_i(n', s)$ die Mengen im Nenner leer sein und es daher zu einer Division durch Null kommen, so setzen wir diese Größen auf $\perp$ (undefiniert). Alle Rechnungen, bei denen $\perp$ als Operand auftritt, sollen per definitionem ebenfalls $\perp$ als Ergebnis haben. Bei Vergleichen gelte $r > \perp$ für alle reellen Zahlen r.

Auch hier lassen sich die Werte der δ_i in einer Tabelle vorberechnen, womit wir uns im nächsten Abschnitt beschäftigen werden. Für die Suche verwenden wir eine angepasste Version des Grundalgorithmus aus Kapitel 3, wie sie Algorithmus 4.2 zeigt. Diese kombinieren wir mit dem Backtracking aus Kapitel 3.5 und den Tricks aus Abschnitt 3.6, die in völliger Analogie anwendbar sind. Auch die heuristische Isomorphieerkennung aus Kapitel 3.7.2 lässt sich übertragen: Wir denken uns in jedem Schritt für den aktuell vorliegenden Vektor x die Bewertungen seiner direkten Nachfolger[11] in eine Multimenge

$$\mathcal{E}_x := \{\{\mathscr{E}(x + e_j) : j \in \{0, 1, \ldots, l-1\}\}\}$$

geschrieben. Mit einer Hashfunktion h wie in 3.7.2 berechnen wir dazu einen Hashwert $h(x) := h(\mathcal{E}_x)$. Es ist klar, dass die Hashwerte $h(x_0)$ und $h(x_1)$ für Vektoren x_0 und x_1, bei denen die zugehörigen Ungleichungssysteme $a - \mathcal{T}x_0 \leq \mathcal{T}x' \leq b - \mathcal{T}x_0$, $\omega^T x' = n - \omega^T x_0$ und $a - \mathcal{T}x_1 \leq \mathcal{T}x' \leq b - \mathcal{T}x_1$, $\omega^T x' = n - \omega^T x_1$ isomorph sind[12], denselben Hashwert besitzen. Umgekehrt müssen Vektoren x_0 und x_1 mit demselben Hashwert natürlich nicht zu isomorphen Gleichungssystemen gehören. Sind die Systeme tatsächlich isomorph, so verhält sich Algorithmus 4.2 (bzw. die um das Backtracking erweiterte Version) in den Teilbäumen unterhalb von x_0 und x_1 identisch. Wurde x_0 also schon betrachtet, so kann in diesem Fall bei Erreichen von x_1 sofort ein Backtracking-Schritt durchgeführt werden, ohne die Gefahr, Lösungen zu übersehen, die beim normalen Ablauf gefunden worden wären. Im Falle der Nichtisomorphie wird dagegen durch einen Backtracking-Schritt bei Erreichen von x_1 tatsächlich ein relevanter Teil des Suchraumes ignoriert. Dies dürfte aber relativ selten passieren. Daher erscheint es aus den schon am Ende von Abschnitt 3.7.2 genannten Gründen ratsam, bei wiederholtem Auftreten desselben Hashwertes vom Vorliegen einer Isomorphie auszugehen und einen Backtracking-Schritt durchzuführen.

4.3.2. Vorberechnung der Tabellenwerte

Die bei Vorschreiben der Automorphismenuntergruppe G auftretenden Bahnlängen sind in der Regel nicht alle gleich. Dies stellt einen wesentlichen Unterschied gegenüber Ab-

[11] Darunter verstehen wir die Vektoren $x' > x$ mit $\mathbb{1}^T x' = \mathbb{1}^T x + 1$.

[12] Wir nennen zwei Ungleichungssysteme *isomorph*, wenn sie durch Zeilen- und Variablenvertauschungen auseinander hervorgehen.

Eingabe : **vector** $a, b \in \mathbb{Z}^k, \omega \in (\mathbb{N}^*)^l$: vgl. System (4.3)
matrix $\mathcal{T} \in \mathbb{Z}^{k \times l}$: vgl. System (4.3)
int n: vgl. System (4.3)
Ausgabe : **vector** $x \in \mathbb{N}^l$ mit $a \leq \mathcal{T}x \leq b$ und $\omega^T x = n$ oder **failed**

```
1  procedure generalizedBaseAlgorithm(...) : vector bzw. failed
2  {
3    vector x ← 0⃗; // starte mit Nullvektor
4    while ω^T x < n do
5      v* ←⊥;
6      foreach v ∈ {0, 1, ..., l − 1} do
7        if v* = ⊥ or E(x + e_v) > E(x + e_{v*}) then
8          v* ← v;
9        end if
10     end foreach
11     if E(x + e_{v*}) ∈ {⊥, 0} then // Abbruch, S(x) ist leer
12       return failed;
13     end if
14     x ← x + e_{v*};
15   end while
16   return x;
17 }
```

Algorithmus 4.2: Verallgemeinerter Grundalgorithmus

schnitt 3.3 dar: Bei der Vorberechnung der Tabellenwerte ϵ_i genügt nicht allein die Kenntnis der i-ten Zeile von $\mathcal{T}$, sondern auch der Vektor der Bahnlängen $\omega = (\omega_0, \omega_1, \ldots, \omega_{l-1})$ muss bekannt sein. Für die konkrete Berechnung sind die Kombinationen aus jedem Zeileneintrag mit dem Wert von ω in derselben Spalte interessant. Um zu beschreiben, in wie vielen Spalten hierbei das Wertepaar $(o, m) \in \mathbb{N} \times \mathbb{Z}$ auftritt, definieren wir:

$$d^i_{o,m} := \left|\{j \in \{0, 1, \ldots, l-1\} : \omega_j = o, \mathcal{T}_{ij} = m\}\right|.$$

Stimmen die Werte $d^i_{o,m}$ und $d^j_{o,m}$ für alle $o \in \mathbb{N}$ und $m \in \mathbb{Z}$ überein, so sind auch die Funktionen δ_i und δ_j identisch und die entsprechenden Tabelleneinträge müssen nur einmal berechnet werden.

Die Berechnung an sich kann so vorgenommen werden: Wir betrachten das Polynom

$$p_i(\xi, \eta) := \prod_{(o,m):d^i_{o,m}>0} \left(\sum_{j=0}^{\lfloor \frac{n}{o} \rfloor} \xi^{oj} \eta^{mj} \binom{j + d^i_{o,m} - 1}{j} \right).$$

Sei c_{jk} der Koeffizient zu $\xi^j \eta^k$. Dann ist c_{jk} gerade die Anzahl der Vektoren x', die

$\omega^T x' = j$ und $\mathcal{T}_{i*} x' = k$ erfüllen. $\epsilon_i(n', s)$ berechnet sich durch

$$\epsilon_i(n', s) = \frac{\sum_{k=a_i-s}^{b_i-s} c_{n'k}}{\sum_{k\in\mathbb{Z}} c_{n'k}}.$$

Der Wert von $n' = n - \omega^T x$ liegt stets zwischen 0 und n, daher kann p_i zur schnelleren Berechnung modulo ξ^{n+1} betrachtet werden. Sind - wie es bei den Kramer-Mesner-Matrizen zu linearen Codes über Kettenringen der Fall ist - alle Einträge von $\mathcal{T}$ nichtnegativ, kann man während des Ausmultiplizierens auftretende Monome $\xi^j\eta^k$ mit $k \geq b_i+1$ durch $\xi^j\eta^{b_i+1}$ ersetzen: Ein Exponent $k > b_i$ bei η bedeutet, dass für einen zum Monom $\xi^j\eta^k$ gehörenden Vektor x' und $x \in \mathbb{N}^l$ beliebig stets $\mathcal{T}_{i*}x' > b_i - \mathcal{T}_{i*}x$ gilt, der genaue Wert von k spielt also keine Rolle. Ist zusätzlich $b_i = \infty$, kann man sogar alle $\xi^j\eta^k$ mit $k \geq a_i$ zu $\xi^j\eta^{a_i}$ zusammenfassen, denn für die zugehörigen Vektoren x' gilt immer $a_i - \mathcal{T}_{i*}x \leq \mathcal{T}_{i*}x'$, unabhängig von x und dem genauen Wert von k.

Beispiel 4.3. Wir wollen einen Schritt des Algorithmus für das System aus Beispiel 4.2 bei der Suche nach einem Code der Länge $n = 20$ mit Minimaldistanz 60 durchführen. Es ist also

$$\mathcal{T} = \begin{pmatrix} 26 & 12 & 22 & 12 & 4 \\ 22 & 12 & 26 & 12 & 4 \\ 24 & 12 & 24 & 16 & 0 \\ 24 & 16 & 24 & 0 & 0 \\ 32 & 0 & 32 & 0 & 0 \end{pmatrix}, a = \begin{pmatrix} 60 \\ 60 \\ 60 \\ 60 \\ 60 \end{pmatrix}, b = \begin{pmatrix} \infty \\ \infty \\ \infty \\ \infty \\ \infty \end{pmatrix} \text{ und}$$
$$\omega^T = \begin{pmatrix} 8 & 4 & 8 & 4 & 1 \end{pmatrix}.$$

Der Rechenweg zur Ermittlung der Polynome $p_i(\xi, \eta)$ befindet sich der besseren Lesbarkeit halber im Anhang B. Die Bewertungen im ersten Schritt lauten:

$$\begin{aligned}
\mathscr{E}((1,0,0,0,0)^T) &= \epsilon_0(12,26)\cdot\epsilon_1(12,22)\cdot\epsilon_2(12,24)\cdot\epsilon_3(12,24)\cdot\epsilon_4(12,32) = \\
&= \frac{16}{16}\cdot\frac{10}{16}\cdot\frac{8}{16}\cdot\frac{3}{16}\cdot\frac{6}{16} = \frac{45}{2048} \approx 2,20\cdot 10^{-2}. \\
\mathscr{E}((0,1,0,0,0)^T) &= \epsilon_0(16,12)\cdot\epsilon_1(16,12)\cdot\epsilon_2(16,12)\cdot\epsilon_3(16,16)\cdot\epsilon_4(16,0) = \\
&= \frac{26}{30}\cdot\frac{26}{30}\cdot\frac{15}{30}\cdot\frac{8}{30}\cdot\frac{3}{30} = \frac{169}{16875} \approx 1,00\cdot 10^{-2}. \\
\mathscr{E}((0,0,1,0,0)^T) &= \mathscr{E}((1,0,0,0,0)^T). \\
\mathscr{E}((0,0,0,1,0)^T) &= \epsilon_0(16,12)\cdot\epsilon_1(16,12)\cdot\epsilon_2(16,16)\cdot\epsilon_3(16,0)\cdot\epsilon_4(16,0) = \\
&= \frac{26}{30}\cdot\frac{26}{30}\cdot\frac{16}{30}\cdot\frac{1}{30}\cdot\frac{3}{30} = \frac{338}{253125} \approx 1,34\cdot 10^{-3}. \\
\mathscr{E}((0,0,0,0,1)^T) &= \epsilon_0(19,4)\cdot\epsilon_1(19,4)\cdot\epsilon_2(19,0)\cdot\epsilon_3(19,0)\cdot\epsilon_4(19,0) = \\
&= \frac{30}{30}\cdot\frac{30}{30}\cdot\frac{2}{30}\cdot\frac{1}{30}\cdot\frac{3}{30} = \frac{1}{4500} \approx 2,22\cdot 10^{-4}.
\end{aligned}$$

Im ersten Schritt wird also die Variable x_0 oder x_2 inkrementiert.

Kapitel 5.

Aufbau einer Codedatenbank

Mit Hilfe der in Kapitel 3 und 4 beschriebenen Algorithmen wurde in Zusammenarbeit mit Michael Kiermaier eine umfassende MySQL-Datenbank erstellt und hierin im Wesentlichen die folgenden Informationen aufbewahrt:

(i) Für die verwendeten Kettenringe wurden die Verknüpfungstafeln für Addition und Multiplikation, ein Erzeugendensystem ihrer Automorphismengruppe und Tabellen mit den Werten des Hamming- und des homogenen Gewichts gespeichert.

(ii) Für Untergruppen G von $\mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$ nach Kapitel 4 wurden - neben einigen Eckdaten wie dem zugrundeliegenden Kettenring R, dem Umriss λ, etc. - ein Erzeugendensystem von G, eine Transversale der Operation von G auf $\overline{\mathcal{V}_\lambda}$ und die symmetrisierte Kramer-Mesner-Matrix $\mathcal{M}_G^\chi$ abgelegt.

(iii) Die Ungleichungssysteme wurden mit Hilfe der Programme *Heurico* und *Solver* gelöst (vgl. Abschnitt 6.1 bzw. 6.2). Über die verwendeten Aufrufparameter wurde ausführlich Protokoll geführt, um Mehrfachsuchen zu vermeiden und das Zustandekommen der Lösungen reproduzieren zu können.

(iv) Für die gefundenen Codes wurden die relevanten Parameter wie etwa Länge, Umriss und Minimaldistanz zusammen mit einer Generatormatrix und dem Gewichtszähler des Codes gespeichert.

Die in Punkt (iii) genannten Programme sind in Kapitel 6 erläutert, für die Punkte (i), (ii) und (iv) soll der genaue Aufbau an dieser Stelle beschrieben werden:

5.1. Einfügen der Ringdaten

Für alle verwendeten Ringe wurden Verknüpfungstafeln mit der am Ende von Abschnitt 1.2 beschriebenen Nummerierung der Elemente gemäß der θ-adischen Entwicklung angelegt. Neben den „klassischen“ Vertretern wie $\mathbb{Z}_{p^n}$ oder den Galoisringen wurden auch einige gemäß Satz 1.16 konstruierte Ringe betrachtet.[1] Da hierbei zueinander isomorphe Kettenringe in unterschiedlichen Darstellungen auftreten können, ist es nützlich, die konstruierten Ringe auf Isomorphie zu testen:

[1] Der kleinste nichtkommutative Ring ist hierbei $\mathbb{F}_4[X,\sigma]/(X^2)$, wobei σ der Frobeniusautomorphismus $x \mapsto x^2$ ist.

5.1.1. Testen von Kettenringen auf Isomorphie

Seien R_1 und R_2 zwei endliche Kettenringe mit $q_{R_1} = q_{R_2}$ und $m_{R_1} = m_{R_2}$. Außerdem sei $\alpha_1 \in R_1$ ein Teichmüllererzeuger der Teichmüllergruppe $\mathcal{A}_1^*$ und θ_1 ein Erzeuger von $\mathrm{Rad}(R_1)$. Wie schon in Bemerkung 1.5 gesehen, ist jeder Isomorphismus σ zwischen R_1 und R_2 bereits anhand der Werte $\tilde{\alpha} := \sigma(\alpha_1)$ und $\tilde{\theta} := \sigma(\theta_1)$ festgelegt und stimmt mit $\chi_{\tilde{\alpha}}^{\tilde{\theta}}$ aus Definition 1.8 überein. Umgekehrt ist aber nicht für jede Wahl von $\tilde{\alpha}$ als Teichmüllererzeuger von R_2 und $\tilde{\theta}$ als Erzeuger von $\mathrm{Rad}(R_2)$ die Bijektion $\chi_{\tilde{\alpha}}^{\tilde{\theta}}$ auch ein Ringisomorphismus. Um festzustellen, ob R_1 und R_2 isomorph sind, kann man nun mit $\tilde{\alpha}$ und $\tilde{\theta}$ die Teichmüllererzeuger von R_2 bzw. die zu θ_2 assoziierten Elemente durchlaufen. Durch Prüfen der Bedingungen

$$\chi_{\tilde{\alpha}}^{\tilde{\theta}}(a+b) = \chi_{\tilde{\alpha}}^{\tilde{\theta}}(a) + \chi_{\tilde{\alpha}}^{\tilde{\theta}}(b) \text{ und } \chi_{\tilde{\alpha}}^{\tilde{\theta}}(ab) = \chi_{\tilde{\alpha}}^{\tilde{\theta}}(a)\chi_{\tilde{\alpha}}^{\tilde{\theta}}(b)$$

für beliebige $a, b \in R_1$ lässt sich nachrechnen, ob die induzierte Fortsetzung zusätzlich auch ein Homomorphismus ist. Falls ja, kann das Verfahren abgebrochen werden und R_1 und R_2 sind als isomorph erkannt. Findet sich kein solches Paar $\tilde{\alpha}$ und $\tilde{\theta}$, so sind R_1 und R_2 nicht isomorph. Die Zahl der zu testenden Bilder $\sigma(\alpha)$ kann im nichtkommutativen Fall[2] mit Hilfe des folgenden Lemmas noch etwas reduziert werden:

Lemma 5.1. *Seien R_1 und R_2 endliche Kettenringe mit Teichmüllererzeugern α_1 und α_2. Sei $\mathcal{A}_1^*$ die von α_1 erzeugte und $\mathcal{A}_2^*$ die von α_2 erzeugte Teichmüllergruppe. Sind R_1 und R_2 isomorph, dann gibt es einen Ringisomorphismus $\sigma : R_1 \to R_2$, der $\mathcal{A}_1^*$ in $\mathcal{A}_2^*$ überführt.*

Beweis. R_1 und R_2 sind isomorph, daher gibt es einen Ringisomorphismus $\rho : R_1 \to R_2$. Sei $\mathcal{A}_3^*$ die vom Teichmüllererzeuger $\rho(\alpha_1)$ erzeugte Teichmüllergruppe. Dann überführt ρ die Teichmüllergruppe $\mathcal{A}_1^*$ in $\mathcal{A}_3^*$. Nach Satz 1.5 sind $\mathcal{A}_2^*$ und $\mathcal{A}_3^*$ in ${R_2}^\times$ zueinander konjugiert, also gibt es $e \in {R_2}^\times$, so dass $\mathcal{A}_2^* = e\mathcal{A}_3^* e^{-1}$. Offensichtlich ist die Abbildung $\tau : R_2 \to R_2, r \mapsto ere^{-1}$ ein Ringisomorphismus auf R_2, also auch die Komposition $\sigma := \tau \circ \rho$. σ überführt $\mathcal{A}_1^*$ in $\mathcal{A}_2^*$, womit die Behauptung gezeigt ist. □

Beim Durchlaufen der Teichmüllererzeuger mit $\tilde{\alpha}$ zum Test von R_1 und R_2 auf Isomorphie kann man sich also auf die Erzeuger einer einzigen Teichmüllergruppe in R_2 beschränken.

Bemerkung 5.1. Es ist davon auszugehen, dass hierfür hinsichtlich der Komplexität deutlich bessere Methoden existieren. Für die im Rahmen dieser Arbeit generierten Kettenringe (alle von Ordnung ≤ 256) erwies sich das geschilderte Verfahren jedoch als völlig ausreichend.

Um gemäß Kapitel 4 Automorphismenuntergruppen für die Codes vorschreiben zu können, muss man die Automorphismengruppe der Kettenringe kennen:

[2] Das Lemma gilt natürlich auch, wenn R_1 und R_2 kommutativ sind, allerdings gibt es dann nach Satz 1.5 nur eine einzige Teichmüllergruppe in R_2 und die Aussage des Lemmas ist trivial.

5.1.2. Bestimmung der Ringautomorphismen

Ringautomorphismen sind natürlich insbesondere Ringisomorphismen, insofern kann man das oben geschilderte Verfahren auch einsetzen, um die Automorphismen eines Kettenringes R zu bestimmen. Die Suche muss lediglich dahingehend verändert werden, dass sie auch nach dem Auffinden des ersten Automorphismus fortgesetzt wird, bis alle $\chi_{\tilde{\alpha}}^{\tilde{\theta}}$ getestet und somit alle Automorphismen erkannt wurden. Allerdings muss man hier in der Regel deutlich weniger Kandidaten prüfen. Grundlage dafür ist das folgende Lemma. Hier und in den weiteren Teilen dieses Kapitels bezeichne $\mathcal{A}^*$ wie gewohnt die von $\alpha \in R$ erzeugte Teichmüllergruppe und die Zahlen p und r mit $q_R = p^r$ seien ebenfalls wie üblich definiert.

Lemma 5.2. *Sei $\sigma \in \mathrm{Aut}(R)$ ein Ringautomorphismus, der $\mathcal{A}^*$ auf sich selbst abbildet. Dann gibt es $s \in \{0, 1, \ldots, r-1\}$, so dass gilt: $\sigma(\alpha) = \alpha^{(p^s)}$.*

Beweis. Sei $\overline{R} := R/\mathrm{Rad}(R)$ und, für $r \in R$, $\overline{r} := r + \mathrm{Rad}(R)$ die Projektion von r auf $\overline{R}$. Dann ist $\overline{\alpha}$ ein primitives Element in $\overline{R}$. Außerdem ist

$$\overline{\sigma} : \overline{R} \to \overline{R}, \overline{r} \mapsto \overline{\sigma(r)}$$

wohldefiniert und damit ein Körperautomorphismus von $\overline{R}$. Die Automorphismengruppe von $\overline{R}$ wird erzeugt vom Frobeniusautomorphismus $\rho : x \mapsto x^p$, daher muss ein $s \in \{0, 1, \ldots, r-1\}$ existieren mit $\overline{\sigma} = \rho^s$. Insbesondere gilt $\overline{\sigma}(\overline{\alpha}) = \overline{\alpha}^{(p^s)} = \overline{\alpha^{(p^s)}}$ und damit $\sigma(\alpha) \equiv \alpha^{(p^s)} \bmod \mathrm{Rad}(R)$. Außerdem muss $\sigma(\alpha)$ nach Voraussetzung in $\mathcal{A}^* = \{\alpha^0, \alpha^1, \ldots, \alpha^{p^r-1}\}$ liegen. Die Elemente aus $\mathcal{A}^*$ sind modulo $\mathrm{Rad}(R)$ paarweise verschieden und nur $\alpha^{(p^s)} \in \mathcal{A}^*$ erfüllt die Kongruenz. Somit gilt $\sigma(\alpha) = \alpha^{(p^s)}$. □

Unser Ziel ist die Bestimmung eines Erzeugendensystems der Automorphismengruppe $\mathrm{Aut}(R)$. Aus diesem können dann mit einem dem Bahnalgorithmus 4.1 sehr ähnlichen Verfahren bei Bedarf die Ordnung bzw. die Elemente von $\mathrm{Aut}(R)$ ermittelt werden. Um die Zahl der zu testenden Kandidaten weiter zu reduzieren, betrachten wir Erzeugermengen bestimmter Untergruppen von $\mathrm{Aut}(R)$, wie sie in der folgenden Definition gegeben werden.

Definition 5.1.

(i) Ringautomorphismen $\sigma \in \mathrm{Aut}(R)$ vom Typ $\sigma(r) = ere^{-1}$ mit $e \in R^\times$, wie sie beispielsweise im Beweis von Lemma 5.1 auftraten, heißen *innere Automorphismen* von R. Die Gruppe aller inneren Automorphismen von R notieren wir mit $\mathrm{Aut}_{\mathrm{inn}}(R)$.

(ii) $S_{\mathcal{A}^*} := \{\sigma \in \mathrm{Aut}(R) : \sigma(\mathcal{A}^*) = \mathcal{A}^*\} \overset{\text{Lemma 5.2}}{=} \{\sigma \in \mathrm{Aut}(R) : \exists s \in \mathbb{N} : \sigma(\alpha) = \alpha^{(p^s)}\}$, der mengenweise Stabilisator von $\mathcal{A}^*$ unter der kanonischen Operation von $\mathrm{Aut}(R)$ auf R.

(iii) $S_\alpha := \{\sigma \in \mathrm{Aut}(R) : \sigma(\alpha) = \alpha\}$, der punktweise Stabilisator von α unter der kanonischen Operation von $\mathrm{Aut}(R)$ auf R.

Lemma 5.3. *Sei $\sigma \in \mathrm{Aut}(R)$ ein Ringautomorphismus. Dann gibt es $\rho \in \mathrm{Aut}_{\mathrm{inn}}(R)$ und $\tau \in S_{\mathcal{A}^*}$, so dass $\sigma = \rho \circ \tau$.*

Beweis. Sei $\mathcal{A}_2^*$ die von $\sigma(\alpha)$ erzeugte Teichmüllergruppe. Mit Lemma 5.1 folgt die Existenz eines inneren Automorphismus $\mu \in \mathrm{Aut}_{\mathrm{inn}}(R)$, der $\mathcal{A}_2^*$ in $\mathcal{A}^*$ überführt. Für $\tau := \mu \circ \sigma$ gilt $\tau \in S_{\mathcal{A}^*}$. Setzt man $\rho := \mu^{-1}$ und wendet ρ auf beide Seiten der Gleichung an, folgt die Behauptung.

□

Lemma 5.4. *Die Menge $\mathfrak{p} := \{\overline{s} \in \mathbb{Z}_r : \exists \sigma \in S_{\mathcal{A}^*} : \sigma(\alpha) = \alpha^{(p^s)}\}$ ist eine Untergruppe von $(\mathbb{Z}_r, +)$ und wird erzeugt von $\overline{u_\mathfrak{p}}$ mit $u_\mathfrak{p} := \min\{s \in \{1, 2, \ldots, r\} : \overline{s} \in \mathfrak{p}\}$.*

Beweis. Wegen $\mathrm{id} \in S_{\mathcal{A}^*}$ und $\mathrm{id}(\alpha) = \alpha^{(p^0)}$ gilt $\overline{0} \in \mathfrak{p}$. Sind $\overline{a}, \overline{b} \in \mathfrak{p}$, so gibt es $\sigma_a, \sigma_b \in S_{\mathcal{A}^*}$ mit $\sigma_a(\alpha) = \alpha^{(p^a)}$ und $\sigma_b(\alpha) = \alpha^{(p^b)}$. Es gilt $\sigma_a \circ \sigma_b \in S_{\mathcal{A}^*}$ und $(\sigma_a \circ \sigma_b)(\alpha) = \alpha^{(p^{(a+b)})}$, also $\overline{a} + \overline{b} \in \mathfrak{p}$. Damit ist $\mathfrak{p}$ als Untergruppe von $(\mathbb{Z}_r, +)$ nachgewiesen. Dass $\mathfrak{p} = \langle \overline{u_\mathfrak{p}} \rangle$ gilt, sieht man so: „$\supset$“ ist klar und gäbe es ein $\overline{v} \in \mathfrak{p} \setminus \langle \overline{u_\mathfrak{p}} \rangle$, dann auch ein $c \in \mathrm{ggT}(u_\mathfrak{p}, v)$ mit $0 < c < u_\mathfrak{p}$. Mit dem erweiterten euklidischen Algorithmus ließen sich dann $x, y \in \mathbb{Z}$ ermitteln, so dass $c = xu_\mathfrak{p} + yv$. Es wäre dann aber auch $\overline{c} = \overline{xu_\mathfrak{p}} + \overline{yv} \in \mathfrak{p}$, ein Widerspruch zur Minimalitätsvoraussetzung für $u_\mathfrak{p}$.

□

Nun haben wir alle Hilfsmittel zusammen, um auf effizientere Art als zuvor ein Erzeugendensystem von $\mathrm{Aut}(R)$ zu berechnen:

Satz 5.5. *Sei A ein Erzeugendensystem von $\mathrm{Aut}_{\mathrm{inn}}(R)$, B ein Erzeugendensystem von S_α und $\mu \in S_{\mathcal{A}^*}$ mit $\mu(\alpha) = \alpha^{(p^{u_\mathfrak{p}})}$. Dann ist $A \cup \{\mu\} \cup B$ ein Erzeugendensystem von $\mathrm{Aut}(R)$.*

Beweis. Sei $\sigma \in \mathrm{Aut}(R)$. Wir zeigen, dass sich σ durch A, B und μ erzeugen lässt: Nach Lemma 5.3 gibt es $\rho \in \mathrm{Aut}_{\mathrm{inn}}(R)$ und $\tau \in S_{\mathcal{A}^*}$ mit $\sigma = \rho \circ \tau$. ρ liegt im Erzeugnis von A, es genügt also zu zeigen, dass τ im Erzeugnis von μ und B liegt. Aufgrund der Definition von μ gibt es $s \in \{0, 1, \ldots, r-1\}$, so dass $\tau(\alpha) = (\mu^s)(\alpha)$. Also gilt $((\mu^s)^{-1} \circ \tau) \in S_\alpha$ und damit liegt τ im Erzeugnis von μ und B.

□

Bemerkung 5.2. Im Rahmen dieser Arbeit wurden die Erzeugendensysteme von $\mathrm{Aut}(R)$ so berechnet:

(i) Ausgehend von $A' = \mathrm{Aut}_{\mathrm{inn}}(R)$ wurden aus A' solange Erzeuger entfernt, bis ein irredundantes Erzeugendensystem A von $\mathrm{Aut}_{\mathrm{inn}}(R)$ erreicht war.

(ii) Im Fall $u_\mathfrak{p} \neq r$ wurde ein $\mu \in S_{\mathcal{A}^*}$ mit $\mu(\alpha) = \alpha^{(p^{u_\mathfrak{p}})}$ bestimmt und zu A hinzugefügt. Zur Berechnung von μ wurde die Menge $\{1, 2, \ldots, r-1\}$ in einer äußeren Schleife mit einer Variable s aufsteigend durchlaufen und die Menge der Erzeuger von $\mathrm{Rad}(R)$ in einer inneren Schleife mit einer Variable $\tilde{\theta}$. Falls die induzierte Abbildung $\chi^{\tilde{\theta}}_{\alpha^{(p^s)}}$ sich als Ringautomorphismus herausstellte, wurde $\mu = \chi^{\tilde{\theta}}_{\alpha^{(p^s)}}$ gesetzt und die Suche beendet. Wurde kein solches $\chi^{\tilde{\theta}}_{\alpha^{(p^s)}}$ gefunden, lag der Fall $u_\mathfrak{p} = r$ vor und A wurde unverändert gelassen.

(iii) Anschließend wurde ein Erzeugendensystem B von S_α bestimmt, indem die Menge der Erzeuger von $\mathrm{Rad}(R)$ mit $\tilde{\theta}$ durchlaufen und jede der Abbildungen $\chi_\alpha^{\tilde{\theta}}$, die als Automorphismus identifiziert werden konnte, in B aufgenommen wurde. Danach wurde die Menge $A \cup B$ durch sukzessives Entfernen von in B liegenden Elementen auf ein irredundantes Erzeugendensystem von $\mathrm{Aut}(R)$ reduziert.

(iv) Die Reduktion der Erzeugendensysteme wurde mit dem Computeralgebrasystem *MAGMA* [Com] mittels Darstellung von $\mathrm{Aut}(R)$ als Permutationsgruppe durchgeführt.

5.2. Generierung der vorgeschriebenen Automorphismengruppen

Für die Konstruktion von Codes mit der Kramer-Mesner-Methode, wie sie in Kapitel 4 beschrieben wird, wurden für zahlreiche Kettenringe R und Umrisse λ Untergruppen G von $\mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$ generiert. Diese wurden in Form eines Erzeugendensystems E in der Datenbank abgelegt und separat auch die zugehörige symmetrisierte Kramer-Mesner-Matrix $\mathcal{M}_G^\chi$ (vgl. Bemerkung 4.4) und die Bahnrepräsentanten zur Operation von G auf $\overline{\mathcal{V}_\lambda}$ gespeichert. Um das mehrfache Ablegen von Matrizen zu isomorphen Systemen zu verhindern, wurde für jede Matrix $\mathcal{M}_G^\chi$ durch iterierte Klassifizierung über die Zeilen und Spalten (siehe Algorithmus 5.1) ein unter Zeilen- und Spaltenvertauschungen invarianter Hashwert $h(\mathcal{M}_G^\chi)$ bestimmt.[3] Da $\mathcal{M}_G^\chi$ direkt aus E konstruiert werden kann, ist es gerechtfertigt, $h(\mathcal{M}_G^\chi)$ auch als den Hashwert von E zu bezeichnen. Von Erzeugendensystemen über demselben Ring mit dem gleichen Hashwert wurde jeweils nur ein einziger Vertreter gespeichert, bei dem die Zahl der Erzeuger minimal war. Außerdem wurde $\mathcal{M}_G^\chi$, sofern dies in akzeptabler Zeit möglich war, mit Hilfe eines von Thomas Feulner entwickelten Programmes kanonisiert, also in eine normierte Form unter allen isomorphen Ausprägungen gebracht. Hierdurch konnte erkannt werden, wenn Erzeugendensysteme über verschiedenen Kettenringen zu äquivalenten Ungleichungssystemen führten. Dies hatte zwei Vorteile: Zum einen musste die normierte Version der Matrix für alle diese Kettenringe nur ein einziges Mal gespeichert werden, zum anderen konnten die Lösungen zu den entsprechenden Ungleichungssystemen ringübergreifend verwendet werden. Eine Lösung zu einem Ungleichungssystem, welches durch Vorschreiben einer Gruppe über dem Ring $\mathbb{Z}_4$ entstand, konnte also ggf. sowohl zu einem entsprechenden Code über $\mathbb{Z}_4$ als auch über $\mathbb{Z}_2[X]/(X^2)$ führen. Reichte die Zeit dagegen nicht zur Kanonisierung aus, wurden die Matrix $\mathcal{M}_G^\chi$ und die Spaltenrepräsentanten in unnormierter Form mit einem entsprechenden Vermerk gespeichert.

Die Generierung der Erzeugendensysteme lief wie folgt ab: Zunächst wurden ein Ring R, ein Umriss λ und ein Wert μ für die maximale Anzahl von Erzeugern der Untergruppen festgelegt, genauso wie ein Wert j_0 für die maximale Spaltenanzahl der Kramer-Mesner-Matrizen und eine Maximalzeit t für die Kanonisierung. Dann wurde, immer

[3] Es war hierbei jedoch nicht ausgeschlossen, dass Matrizen zu nichtisomorphen Systemen den gleichen Hashwert erhielten.

ausgehend von $E := \emptyset$, das folgende rekursive Verfahren solange wiederholt, bis eine zufriedenstellende Anzahl an Gruppen gefunden war:

(i) Berechne die Kramer-Mesner-Matrix $\mathcal{M}_G^\chi \in \mathbb{N}^{i \times j}$ zu der von E generierten Gruppe G. Falls $j > j_0$, überspringe (ii) und gehe direkt zu (v). Ist $E \neq \emptyset$ und hat sich die Spaltenanzahl von $\mathcal{M}_G^\chi$ gegenüber dem Vorgängerschritt nicht verkleinert, gehe in der Rekursion einen Schritt zurück.

(ii) Berechne den Hashwert $h(\mathcal{M}_G^\chi)$. Teste, ob $h(\mathcal{M}_G^\chi)$ bereits für ein Erzeugendensystem E' über dem Ring R in der Datenbank eingetragen ist. Falls nein, gehe zu (iv).

(iii) Prüfe, ob $|E| < |E'|$. Falls nein, gehe zu (v). Sonst teste, ob die zu E' gehörende Matrix kanonisiert ist. Falls nein, gehört die Matrix auch nicht zu einem Erzeugendensystem über einem anderen Ring und kann daher zusammen mit E' und den Spaltenrepräsentanten gelöscht werden. Sonst lösche nur E' und die Spaltenrepräsentanten.

(iv) Speichere E in der Datenbank. Versuche, $\mathcal{M}_G^\chi$ innerhalb der Zeit t zu kanonisieren.

  (a) Falls dies gelang, speichere die entsprechend untereinander permutierten Spaltenrepräsentanten zur kanonisierten Version $\overline{\mathcal{M}_G^\chi}$ in der Datenbank. Prüfe, ob $\overline{\mathcal{M}_G^\chi}$ selbst dort bereits vorhanden ist. Falls nein, speichere $\overline{\mathcal{M}_G^\chi}$ in der Datenbank. Erstelle nun einen Verweis von E auf den Eintrag zu $\overline{\mathcal{M}_G^\chi}$.

  (b) Wurde die Kanonisierung dagegen nicht rechtzeitig beendet, speichere $\mathcal{M}_G^\chi$ und die Spaltenrepräsentanten mit einem entsprechenden Vermerk in der unkanonisierten Form. Erstelle einen Verweis von E auf $\mathcal{M}_G^\chi$.

(v) Falls $|E| = \mu$, gehe einen Schritt in der Rekursion zurück. Sonst ermittle zufällig[4] ein Element $x \in \mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$. Bestimme die Ordnung o von x. Nun durchlaufe mit t die Menge der Teiler von o mit $t < o$. Springe dabei jeweils mit $E \leftarrow E \cup \{x^t\}$ in den nächsten Rekursionsschritt, also zu (i). Sind alle solchen Teiler t durchlaufen, gehe in der Rekursion einen Schritt zurück.

5.3. Suche nach Codes

Für die Suche nach Codes mit beliebiger Automorphismengruppe wie in Kapitel 3 kam das Programm *Heurico* (vgl. Abschnitt 6.1) zum Einsatz. Die Kramer-Mesner-Systeme aus Kapitel 4 wurden mit dem Programm *Solver* (vgl. Abschnitt 6.2) gelöst.

[4] Im Allgemeinen dürfte es sehr schwer sein, Elemente gleichverteilt über eine nur durch ein Erzeugendensystem gegebene Gruppe auszuwählen; an dieser Stelle kam daher lediglich ein näherungsweise zufälliges Verfahren zum Einsatz.

5.3.1. Suche mit Heurico

Die meisten mit Hilfe von *Heurico* konstruierten Codes wurden unter Verwendung des Schalters *-autosearchcodes* gefunden. Zu einem vorgegebenen Kettenring R, einer Maximallänge n_0, einem $s \in \mathbb{N}^*$ und einer maximalen Suchzeit t erfolgte damit automatisiert eine Suche nach Codes für alle Umrisse $\lambda = (\lambda_0, \ldots, \lambda_{k-1}) \vdash s$ mit Längen zwischen k und n_0. Die Bestimmung der unteren Schranke für die Minimaldistanz d erfolgte dabei dynamisch: Der Startwert wurde, sofern möglich, durch die Minimaldistanz des besten gefundenen Codes mit um eins kürzerer Länge und gleichem Umriss initialisiert. Immer, wenn eine Suche erfolgreich war, wurde sie noch einmal mit um eins erhöhter Minimaldistanz durchgeführt. Erfolgte dagegen ein Abbruch, weil die Maximalzeit überschritten war, so wurde ein neuer Versuch mit aggressiveren Parametern[5] für die Heuristik unternommen. Dem lag die Idee zugrunde, dass ein stärker ausgedünnter Suchbaum in derselben Zeit globaler - wenngleich dementsprechend auch oberflächlicher - untersucht wird. Für genauere Details sei hier auf die Beschreibung des Schalters *-autosearchcodes* in Abschnitt 6.1.3 verwiesen. Der Startwert für den Parameter a lag in den Suchläufen jeweils bei $a = 1.5$ oder $a = 3.0$.

5.3.2. Suche mit Solver

Vor der Konstruktion von linearen Codes zum Kettenring R vom Umriss λ mit Hilfe der Kramer-Mesner-Methode wurden zunächst passende Gruppen auf die in 5.2 beschriebene Weise generiert. In den größeren Fällen blieb diese Suche aufgrund der schnell ausufernden Zahl an möglichen Untergruppen von $\mathcal{A}^\lambda \rtimes \mathrm{Aut}(R)$ häufig auf die zyklischen beschränkt. Für die Kanonisierung der Matrizen wurde in der Regel eine Maximalzeit von 60 Sekunden eingeräumt. Die Konstruktion der Codes erfolgte auch hier weitestgehend automatisiert: Von außen wurde ein Kettenring R, eine Maximallänge n_0, ein $s \in \mathbb{N}^*$ und eine maximale Suchzeit t vorgegeben. Mit Hilfe eines Skripts wurden dann, mit aufsteigendem n, Suchen nach Codes der Länge $n \leq n_0$ über R für alle Gruppen zu einem geeigneten Umriss $\lambda = (\lambda_0, \ldots, \lambda_{k-1}) \vdash s$ durchgeführt. Da die mit dem Programm *Heurico* erzielten Ergebnisse bereits vorlagen, konnte die dort gefundene untere Schranke d' für die Minimaldistanz meist zur Initialisierung von d vermöge $d := d' + 1$ verwendet werden. Existierte kein solcher Referenzwert, wurde d ggf. so initialisiert, dass das (in der Regel nicht lineare) Graybild eines Codes über R mit Minimaldistanz d die in [Gra] aufgeführten unteren Schranken für entsprechende lineare Codes übertroffen hätte. Gab es auch diesbezüglich keine Vergleichsdaten, wurde d auf den besten Wert aus der Suche mit um eins kleinerer Länge oder - falls eine solche nicht erfolgt war - auf eins gesetzt. Wie bei den Läufen mit *Heurico* wurde die Suche über die Gruppen im Erfolgsfall mit $d = d + 1$ bei unverändertem n wiederholt.

[5] Das sind solche, die zu einer stärkeren Reduktion des Suchbaumes führen.

```
   Eingabe : matrix M = (m_ij) ∈ Z_N^{r×c} (N ist maximal darstellbare Zahl)
   Ausgabe : int h: Hashwert von M
 1 int rowHash[0..r − 1]; // globales Array für Zeilenhashes
 2 int colHash[0..c − 1]; // globales Array für Spaltenhashes
 3 procedure iteratedClassification(...) : int
 4 {
 5   rowHash[0..r − 1] ← 0; // initialisiere rowHash mit 0
 6   colHash[0..c − 1] ← 0; // initialisiere colHash mit 0
 7   int s ← 1; // s ist Zahl verschiedener Zeilenhashes
 8   int t ← 0; // t speichert vorherigen Wert von s
 9   while s ≠ t do // s = t bedeutet Stagnation
10       t ← s;
11       rowHashes2colHashes(); // neue Zeilenhashes aus Spaltenhashes
12       s ← colHashes2rowHashes(); // neue Spaltenhashes aus Zeilenhashes
13   end while
14   return mixHashes(rowHash[], colHash[]);
15 }
16 procedure colHashesToRowHashes() : int
17 {
18   set S ← ∅; // S ist Menge verschiedener Zeilenhashes
19   for int i from 0 to r − 1 do
         // Bestimme nun Fingerabdruck der Zeile i
         rowHash[i] ← pairHash(m_i0, colHash[0]);
20       for int j from 1 to c − 1 do
21           rowHash[i] ← rowHash[i] ∗ pairHash(m_ij, colHash[j]);
22       end for
23       S ← S ∪ rowHash[i];
24   end for
25   return |S|;
26 }
27 procedure rowHashesToColHashes() : int
28 {
29   analog zu colHashesToRowHashes();
30 }
```

Algorithmus 5.1: Iterierte Klassifizierung von Matrizen: pairHash : $\mathbb{Z}_N^2 \to \mathbb{Z}_N$ stellt idealerweise eine strukturlose Surjektion dar, während $* : \mathbb{Z}_N^2 \to \mathbb{Z}_N$ ein sowohl assoziativer als auch kommutativer Operator ist, dessen Werte gleichmäßig über $\mathbb{Z}_N$ streuen. Ferner sei mixHashes eine Funktion, die aus zwei Arrays von Hashwerten einen einzigen Hashwert berechnet; sie sollte konstant unter Vertauschung der Wertereihenfolge innerhalb der beiden Arrays sein, aber ansonsten keine Struktur aufweisen. Da Kramer-Mesner-Matrizen in der symmetrisierten Version (durch entsprechende Darstellung der Periodentupel) ebenfalls als Matrizen über $\mathbb{Z}_N$ interpretiert werden können, ist der Algorithmus auch hierfür anwendbar.

Kapitel 6.

Entwickelte Programme

6.1. Das Programm Heurico

Heurico wurde geschrieben für die Suche nach linearen Codes über endlichen Kettenringen *ohne* vorgeschriebene Automorphismengruppe. Durch entsprechende Wahl der Aufrufparameter kann es aber auch zur Suche nach Arcs verwendet werden. Es basiert im Wesentlichen auf den in Kapitel 3 vorgestellten Ideen und Algorithmen. Für die Implementierung der Kettenringarithmetik kam eine von Michael Kiermaier ins Leben gerufene und gemeinsam mit dem Autor dieser Arbeit weiterentwickelte Template-Bibliothek für C++ zum Einsatz. Die hiervon benötigten Teile befinden sich auf der beigefügten CD im Verzeichnis `heurico2/Ringe`. Ferner werden die *Boost C++ Libraries* eingebunden.

6.1.1. Kompilierung und Installation

Der Quellcode für *Heurico* befindet sich auf der beigefügten CD im Ordner `heurico2`. Er sollte sich unter den Betriebssystemen *Windows* und *Linux* mit jedem aktuellen Compiler für C++ übersetzen lassen.[1] Wir beschreiben hier den Kompiliervorgang mit der *GNU Compiler Collection* (gcc) unter dem Betriebssystem *Linux*. Alle Eingaben erfolgen dabei auf der Konsole.

(i) Voraussetzung für eine erfolgreiche Kompilierung ist, dass die *GNU Compiler Collection* und die *Boost C++ Libraries* installiert sind. Sollte das noch nicht der Fall sein, so geschieht dies in der Regel - abhängig von der Linux-Distribution - mit den Paketverwaltungssystemen *RPM* oder *APT*. Unter *APT* beispielsweise lauten die Aufrufe für gewöhnlich:

```
sudo apt-get install g++ und
sudo apt-get install libboost-dev
```

(ii) Mit dem `cd`-Kommando in das Verzeichnis `heurico2` wechseln und dort `make` eingeben.

(iii) Das Programm sollte nun fehlerfrei übersetzt werden. Je nach Compiler-Version erscheinen eventuell Warnungen, diese sind jedoch harmlos.

[1] Das Makefile ist allerdings speziell auf die Kompilierung mit `gcc` unter *Linux* zugeschnitten; außerdem darf bei Übersetzung unter Windows der Präprozessorschalter `LINUX` nicht gesetzt sein.

(iv) Die so erstellte ausführbare Datei heißt `heurico2` und befindet sich im Ordner `bin.release`. Sie kann vom aktuellen Ordner aus mit

```
bin.release/heurico2 PARAMETER
```

gestartet werden. Dabei ist `PARAMETER` durch die Suchparameter zu ersetzen, mit denen man *Heurico* starten möchte. Sie werden im folgenden Abschnitt genauer beschrieben.

6.1.2. Aufrufkonvention und Ausgabe

Der Standardaufruf von *Heurico* soll an einem Beispiel erklärt werden. Wir wollen das Programm den *Oktacode* konstruieren lassen, das $\mathbb{Z}_4$-lineare Urbild des bekannten *Nordstrom-Robinson-Codes*. Er besitzt die Länge $n = 8$ und die homogene Minimaldistanz $d_{\text{hom}} = 6$. Sein Umriss ist gegeben durch $\lambda = (2, 2, 2, 2)$. Wir gehen davon aus, dass die im vorherigen Abschnitt beschriebene Installation problemlos abgeschlossen wurde und die Eingabekonsole auf dem Verzeichnis `heurico2` der CD steht. Um den Oktacode zu suchen, geben wir folgende Zeile (natürlich ohne den hier aus Platzgründen erfolgten Umbruch) ein:

```
bin.release/heurico2 -ring Ringe/Kettenringe/004-002-1-2-Z4 -n 8 -k 4 2
2 2 2 -d 6 -homweight
```

Hier eine Erklärung der einzelnen Bestandteile des Aufrufs:

- `-ring Ringe/Kettenringe/004-002-1-2-Z4`: Definiert den Ring, über dem der Code konstruiert werden soll. Der zweite Teil ist dabei der Name einer Datei, in dem die Daten des Rings, wie z. B. der Name, die charakteristischen Parameter und die Verknüpfungstafeln für Addition und Multiplikation gespeichert sind. Auch die Festlegung von α und θ findet hier statt: Falls $q \neq 2$, so ist α das Element mit der Nummer 2 in der Datei, ansonsten das Element mit der Nummer 1, welches stets die multiplikative Einheit darstellt. θ wird, falls $m > 1$, durch das Elemente mit der Nummer q repräsentiert, sonst ist R ein Körper und $\theta = 0$, also das neutrale Element der Addition, welches immer die Nummer 0 besitzt. Im Verzeichnis `heurico2/Ringe/Kettenringe` befinden sich etliche weitere Dateien für Kettenringe kleiner Ordnung.

- `-n 8`: Definiert die Länge des zu konstruierenden Codes, in diesem Fall $n = 8$.

- `-k 4 2 2 2 2`: Legt den Umriss des Codes fest. Der erste Wert nach `-k` gibt die Zeilenanzahl der Generatormatrix an, hierauf werden der Reihe nach die Komponenten des Umrisses aufgelistet.

- `-d 6`: Die Minimaldistanz, nach der gesucht werden soll. Es handelt sich hierbei um eine untere Schranke, daher ist prinzipiell nicht ausgeschlossen, dass am Ende ein Code mit höherer Minimaldistanz konstruiert wird. Dies dürfte in der Praxis allerdings nur selten vorkommen.

- `-homweight`: Legt fest, dass sich die angegebene Minimaldistanz bzw. das Minimalgewicht des Codes auf das homogene Gewicht beziehen. Die Skalierung ist dabei die in Definition 2.7 gegebene. Die Alternativangabe an dieser Stelle lautet `-hamweight` und legt das Hamminggewicht (skaliert auf die Werte 0 und 1) zugrunde.

Nach Eingabe der obigen Zeile erscheinen, neben einigen anderen Informationen zum Suchvorgang, die folgenden Ausgaben:

- `Found a solution after ... seconds`: Gibt an, dass eine Lösung gefunden wurde, zusammen mit der benötigten Zeit in Sekunden. Die Ausgabe im Falle eines Scheiterns lautet `Search failed!`.

- `maxBranchDiff = 1.00`: Dies ist - vereinfacht dargestellt - der maximale Wert für den Suchparameter a aus Algorithmus 3.3, mit dem die Lösung gerade noch gefunden worden wäre. Er wird ermittelt durch Maximierung entlang des zur Lösung führenden Pfades über die Quotienten aus der bestbewerteten und der für die Lösung gewählten Variable. Da im Beispiel keinerlei Backtracking erfolgte, ist der Wert hier 1.00.

- `maxBranchNumber = 1`: Das Maximum unter den Ordinalzahlen der gewählten Variablen entlang des Lösungspfades. Der Wert 1 bedeutet, dass stets die bestbewertete Variable gewählt wurde.

- `weight enumerator: w(x) = 1x^0+112x^6+30x^8+112x^10+1x^16`: Gibt den Gewichtszähler des Oktacodes an.

- `generator matrix: (...)`: Gibt eine Generatormatrix des Oktacodes an.

- `nodes = 8`: Die Zahl der besuchten Knoten im Suchbaum. Jeder Knoten korrespondiert mit genau einem Vektor $x^{(i)}$ in Algorithmus 3.3. In Falle des Beispiels gibt es für jeden Zwischenschritt von $\mathbb{1}^T x^{(i)} = 0$ bis $\mathbb{1}^T x^{(i)} = 7$ genau einen Knoten.

- `evaluations = 423`: Bedeutet, dass insgesamt 423 Mal die Funktion $\mathcal{E}$ aus Abschnitt 3.2 aufgerufen wurde.

- `hash cuts = 0`: Die Zahl der Abschneidungen aufgrund der heuristischen Isomorphieerkennung nach 3.7. Im Beispiel kam sie mangels Backtracking nicht zum Einsatz, daher der Wert 0.

6.1.3. Zusätzliche Aufrufparameter

Heurico unterstützt neben den bereits zuvor genannten noch einige weitere Aufrufparameter. Die wichtigsten sollen hier kurz beschrieben werden:

- `-a <maxquot>`: Setzt den Parameter a aus Algorithmus 3.3 auf den Wert `maxquot`.

- `-w <w>`: Setzt die maximale Ordinalzahl w, bis zu der Variablen innerhalb eines Knotens als Alternative zur bestbewerteten untersucht werden, auf `w`. Ist beispielsweise $w = 5$, so werden innerhalb eines Knotens höchstens die besten fünf Variablen weiterverfolgt.[2]

- `-t <timeout>`: Bricht die Suche ab, wenn nach `timeout` Sekunden noch keine Lösung gefunden wurde.

- `-arcs <k>`: Sucht anstelle von Codes nach Arcs in $\mathrm{PHG}(k-1, R)$. Als zusätzliche Parameter sind `-ring`, `-n` und `-d` anzugeben. Wie in Abschnitt 2.4 beschrieben, wird hiermit $\lambda = (m, m, \ldots, m)$ gesetzt, die Mengen $\overline{\mathcal{V}_\lambda}$ und $\overline{\mathcal{U}_\lambda}$ durch $\overline{\mathcal{V}_\lambda{}^f}$ bzw. $\overline{\mathcal{U}_\lambda{}^f}$ ersetzt und das Hamminggewicht verwendet. Gesucht wird dann nach einem $(n, n-d)$-Arc in $\mathrm{PHG}(k-1, R)$.

- `-autosearchcodes`: Dient zur automatischen Suche nach möglichst guten Codes für verschiedene Umrisse und Längen. In einer äußeren Schleife werden dabei, beginnend mit dem durch den Schalter `-k` vorgegebenen $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{k-1}) \vdash l$, alle weiteren Umrisse, die Partitionen von l darstellen, in lexikographisch absteigender Reihenfolge durchlaufen. In einer inneren Schleife läuft n, ausgehend von dem durch `-n` gegebenen, bis zu dem durch `-upper_n` (siehe unten) eingestellten Wert n_0. Die gesuchte Minimaldistanz d wird zu Anfang durch den Schalter `-d` bestimmt. In den Folgeschritten wird d immer mit dem besten gefundenen Wert für die vorherige Länge initialisiert. Genau dann, wenn eine Suche erfolglos beendet wird, ohne dass die aufgrund eines Überschreitens der durch `-timeout` vorgegebenen Zeit passiert wäre, wird der Durchlauf der inneren Schleife abgeschlossen. Erfolgt der Abbruch dagegen zeitbedingt, so werden sukzessive die mit `-a` und `-w` gesteuerten Werte abgesenkt bzw. der Wert von d erhöht. Die Idee der erstgenannten Maßnahme ist, mit den aggressiveren Abschneidekriterien in der gleichen Zeit mehr verschiedene Teile des Suchbaums betrachten zu können. Das Erhöhen von d mag auf den ersten Blick wenig sinnvoll erscheinen, denn eine Suche nach Minimaldistanz $d+1$ ist in der Regel natürlich schwieriger als eine nach Minimaldistanz d. Allerdings ist der Suchbaum für höhere Werte von d oft deutlich kleiner, da Irrwege viel schneller erkannt werden können. Die Erfahrung hat gezeigt, dass auf diese Weise gelegentlich tatsächlich Codes mit höherer Minimaldistanz als der ursprünglich angesetzten gefunden wurden, die ansonsten aufgrund des Zeitlimits unentdeckt geblieben wären.[3] Abbildung 6.1 zeigt den schematischen Ablauf von *Heurico* bei Aufruf mit `-autosearchcodes`: Bei der Suche im Rahmen dieser Arbeit wurde der Schalter `-w` nicht verwendet (das entspricht dem Startwert $w = \infty$) und die im Flussdiagramm aufgeführten Funktionen $f_w(c)$, $f_a(c, a)$ und $f_d(c, d)$ waren wie folgt definiert:

[2] Weniger sind möglich aufgrund des Einflusses des Parameters a im Backtracking.

[3] Ein Beispiel ist der Code der Länge 21 mit homogener Minimaldistanz 60 zum Umriss $\lambda = (2, 2, 2)$ über $\mathrm{GR}(16, 4)$.

Für $c < 4$:

$$f_w(0) = 8,\ f_w(1) = 5,\ f_w(2) = 3,\ f_w(3) = 2;$$
$$f_a(c, a) = \sqrt{a};$$
$$f_d(c, d) = d.$$

Für $c \geq 4$:

$$f_w(c) = 2;$$
$$f_a(c, a) = a;$$
$$f_d(c, d) = d + 1.$$

- `-ehs <size>`: Begrenzt die maximale Größe der Hashtabelle für die Isometrieerkennung nach Kapitel 3.7 auf `size` Bytes. Bei Überschreiten der Grenze wird dann vor jedem Neueintrag ein alter Wert gelöscht.
- `-chs <size>`: Wie zuvor, aber für eine andere Hashtabelle: Neben derjenigen für die heuristische Isometrieerkennung führt *Heurico* auch noch eine Tabelle mit Hashwerten zu den bisher im Rahmen von Algorithmus 3.3 untersuchten Vektoren $x^{(i)}$. Dies ist sinnvoll, da derselbe Vektor durch unterschiedliche Reihenfolgen bei der Variableninkrementierung erreicht werden kann. Im Gegensatz zur heuristischen Variante ist in diesem Fall eine Suchbaumabschneidung vollkommen sicher.[4] Natürlich würde hier insbesondere auch die heuristische Isometrieerkennung anschlagen, allerdings müssten dafür zunächst die Werte von $\mathcal{E}$ für alle möglichen Vektoren, die aus $x^{(i)}$ durch Inkrementierung einer Komponente hervorgehen, bestimmt werden. In der zweiten Variante dagegen kann die Wiederholung schon bei der Wahl im Vorgängerknoten bemerkt und damit wertvolle Rechenzeit gespart werden.
- `-sol <solfile>`: Schreibt im Erfolgsfall die Generatormatrix zur Lösung, zusammen mit einigen Suchdaten, in die Datei `solfile`.
- `-upper_n <maxn>`: Setzt den Wert n_0 für den Schalter `-autosearchcodes` auf `maxn`.

Bemerkung 6.1. Fast alle von *Heurico* konstruierten Codes wurden unter Verwendung des Schalters `-autosearchcodes` gefunden.

6.2. Das Programm Solver

Obwohl das Programm *Solver* mit der speziellen Intention entwickelt wurde, unter Verwendung der Kramer-Mesner-Methode nach linearen Codes über endlichen Kettenringen zu suchen (vgl. Kapitel 4), lässt es sich ganz allgemein für das Lösen von Diophantischen Ungleichungssystemen einsetzen, welche der Form (4.3) aus Abschnitt 4.3 entsprechen[5].

[4] Denkbar sind selbstverständlich immer noch Fehler durch Hashkollisionen.

[5] Hierunter fallen beispielsweise die Kramer-Mesner-Systeme zu *Designs*, *Networkcodes* und *covering-Codes*.

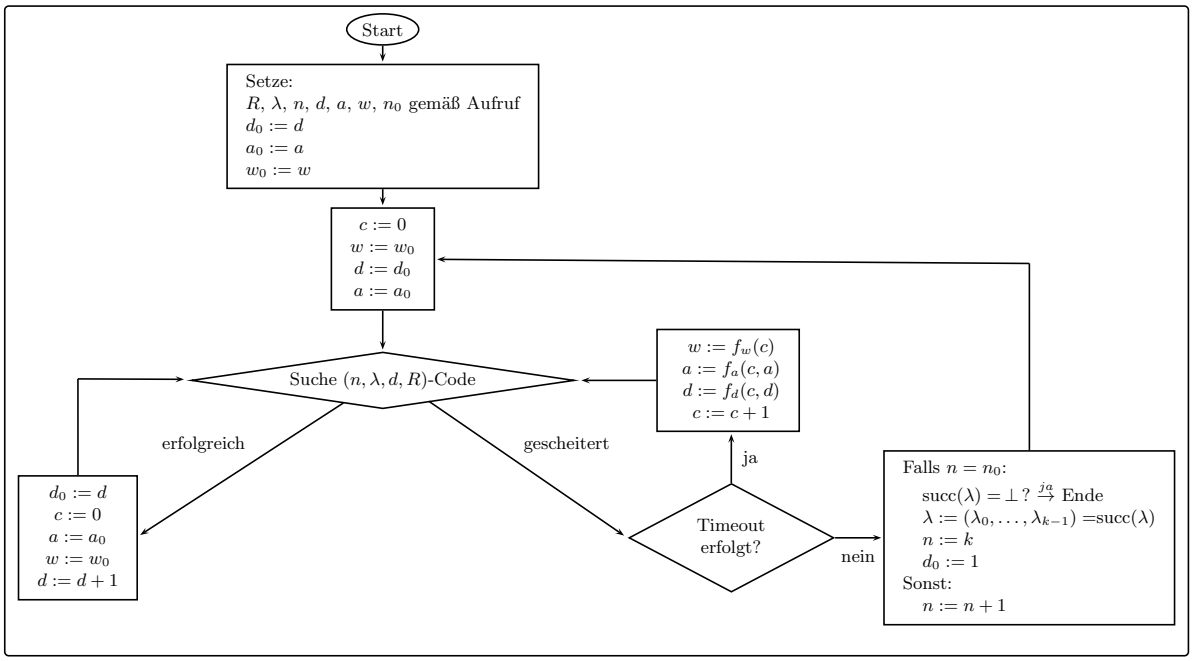

Abbildung 6.1.: Schematischer Ablauf von *Heurico* bei Start mit dem Schalter `-autosearch`. succ(λ) bezeichnet hierbei die Funktion, die den lexikographisch nächstkleineren Umriss liefert, der dieselbe Zahl wie λ partitioniert. Existiert ein solcher nicht, wird $\perp$ zurückgegeben und die Suche terminiert.

6.2.1. Kompilierung und Installation

Der Ordner `solver` der CD enthält die Sourcedateien von *Solver*. Wie bei *Heurico* auch, beschreiben wir nur die Kompilierung mit `gcc` unter *Linux*, obwohl auch unter *Windows* mit aktuellen Compilern keine Probleme zu erwarten sind:[6]

(i) Mit dem `cd`-Kommando in das Verzeichnis `solver` wechseln und dort `make` eingeben. Das Programm sollte nun fehlerfrei übersetzt werden.

(ii) Die so erstellte ausführbare Datei heißt `solver4bpe` und befindet sich im Ordner `bin.release4bpe`. Sie kann vom aktuellen Ordner aus mit

```
bin.release4bpe/solver4bpe PARAMETER
```

gestartet werden.

Bevor wir näher auf die Aufrufkonvention von *Solver* und optionale Argumente eingehen, soll noch eine Kompilierungsvariante erläutert werden: Übersetzt man die Quelldateien mit

[6]Das Makefile wurde wieder für die Kompilierung mit `gcc` unter *Linux* geschrieben, unter *Windows* darf der Präprozessorschalter `LINUX` nicht gesetzt sein.

`make BPE=1` bzw. `make BPE=2`,

so werden die Programmdateien `solver1bpe` bzw. `solver2bpe` erstellt. In diesen Varianten verwendet *Solver* pro Matrixeintrag im Speicher nur ein bzw. zwei Byte[7]. Das ist allerdings nur bei sehr großen Problemen nötig, die sonst nicht mehr in den Arbeitsspeicher des Rechners passen würden. Im Gegenzug sind die Matrixeinträge selbst dann auf das Intervall $[-128, 127]$ bzw. $[-32768, 32767]$ beschränkt. Stößt das Programm auf größere Werte, wird mit einer entsprechenden Fehlermeldung abgebrochen.

6.2.2. Aufrufkonvention und Ausgabe

Der Standardweg, ein Ungleichungssystem an *Solver* zu übergeben, führt über eine Textdatei, die nach folgendem Schema aufbaut ist:

- In der ersten Zeile stehen die Zeilen- und Spaltenanzahl von $\mathcal{T}$, getrennt durch ein oder mehrere Leerzeichen.
- Danach folgen die einzelnen Zeilen von $\mathcal{T}$, getrennt durch mindestens ein Leerzeichen. Am Ende jeder Zeile werden, nach einem oder mehr vorangehenden Leerzeichen, die Werte a_i und b_i als Intervall angegeben, also in der Form $[a_i, b_i]$. Ist $b_i = \infty$, so wird statt einer Zahl `INF` angegeben. Für die untere Intervallgrenze ist dies nicht vorgesehen.
- Für die letzte Gleichung wird genauso verfahren, hier lautet das Intervall auf der rechten Seite $[n, n]$[8]. Die Verwendung von `INF` ist hier nicht möglich.

Die Darstellung von Beispiel 4.3 liest sich dann so:

```
5 5
26 12 22 12 4 [60,INF]
22 12 26 12 4 [60,INF]
24 12 24 16 0 [60,INF]
24 16 24  0 0 [60,INF]
32  0 32  0 0 [60,INF]
 8  4  8  4 1 [20,20]
```

Eine entsprechende Datei ist unter dem Namen `example` im Verzeichnis `solver` der CD enthalten. Das System soll nun gelöst werden: Dazu wechseln wir mit `cd` in eben dieses Verzeichnis und geben

```
bin.release4bpe/solver4bpe -file example -u 0
```

ein. Binnen kürzester Zeit sollte das Programm erfolgreich durchlaufen. Wir wollen einige Ausgaben erläutern:

[7] Standardmäßig werden vier Byte benutzt. Die Abkürzung „bpe“ steht für „bytes per entry“.

[8] *Solver* erlaubt auch, hier echte Intervalle $[n_l, n_u]$ zu verwenden. Die notwendigen Anpassungen im Algorithmus beschränken sich im Wesentlichen darauf, jede Bedingung der Art $\omega^T x = n$ durch $\omega^T x \in [n_l, n_u]$ zu ersetzen.

- `solution vector...`: In der nachfolgenden Zeile wird der gefundene Lösungsvektor aufgelistet, in diesem Fall `1 1 1 0 0`.

- `maxDiff = 1`: Dies ist - vereinfacht dargestellt - der maximale Wert für den Suchparameter a aus Algorithmus 3.3, mit dem die Lösung gerade noch gefunden worden wäre. Er wird ermittelt durch Maximierung entlang des zur Lösung führenden Pfades über die Quotienten aus der bestbewerteten und der für die Lösung gewählten Variable. Da im Beispiel keinerlei Backtracking erfolgte, ist der Wert hier 1.

- `maxBranch = 1`: Das Maximum unter den Ordinalzahlen der gewählten Variablen entlang des Lösungspfades. Der Wert 1 bedeutet, dass stets die bestbewertete Variable gewählt wurde.

- `nodes = 3`: Die Zahl der besuchten Knoten im Suchbaum. Jeder Knoten korrespondiert mit genau einem Vektor $x^{(i)}$ in Algorithmus 3.3.

- `Examining...at step = 0: col = 0...eval = exp(-3.81796)...`: In gewissen Abständen berichtet *Solver* darüber, an welcher Stelle im Suchbaum er sich gerade befindet. Im konkreten Fall geht es um den allerersten Schritt, in dem am Wurzelknoten (`step = 0`) die Inkrementierung der Variable zur ersten Spalte (`col = 0`) ausprobiert wird. Der Wert $-3.81796 \approx \ln(\frac{45}{2048})$ (vgl. die Rechnung zu Beispiel 4.3) ist der natürliche Logarithmus der Bewertungsfunktion $\mathscr{E}$.

Bemerkung 6.2. Die Datei `example_60_13_24_2` zeigt ein größeres Beispiel zur Suche nach einem (optimalen) linearen binären $(60, 2^{13}, 24)$-Code.[9]

6.2.3. Zusätzliche Aufrufparameter

Der Standardaufruf von *Solver* ist `bin.release4bpe/solver4bpe -file PROBLEMFILE`, wo `PROBLEMFILE` der Pfad zur Datei ist, in der das Ungleichungssystem gespeichert ist. Damit startet das Programm mit in der Regel sinnvollen Standardwerten. Suchvorgang und Ausgabeverhalten lassen sich aber durch die folgenden Optionen beeinflussen:

- `-nthreads <t>`: Startet eine parallele Suche mit t Threads, so dass auch Systeme mit mehreren Rechenkernen ausgenutzt werden können.

- `-splitlimit <l>`: Nur sinnvoll in Kombination mit `-nthreads`. Die Option sorgt dafür, dass Knoten im Suchbaum in einem Abstand von mindestens `l` von der Wurzel nicht mehr über mehrere Threads verteilt werden. Ziel ist es, den dabei entstehenden Verwaltungsaufwand zu begrenzen.

- `-a <maxquot>`: Setzt den Parameter a aus Algorithmus 3.3 auf den Wert `maxquot`.

[9] Hier wurde die Kramer-Mesner-Methode auf das System $(E - \mathcal{M}_G^{w_{\mathrm{Ham}}})x \leq (n-d) \cdot \mathbb{1}$, $\mathbb{1}^T x = n$ angewendet, wobei E dieselben Dimensionen wie $\mathcal{M}_G^{w_{\mathrm{Ham}}}$ und nur Einsen als Einträge besitzt. Dieses ist bei Verwendung des Hamminggewichtes über Körpern äquivalent zu $\mathcal{M}_G^{w_{\mathrm{Ham}}} x \geq d \cdot \mathbb{1}$, $\mathbb{1}^T x = n$.

- `-b <u>`: Beschränkt die Größe aller beteiligten Variablen auf `u`. Für von linearen Codes herrührende Systeme kann man mit `-b 1` beispielsweise nach projektiven Codes suchen.

- `-w <w>`: Setzt die maximale Ordinalzahl w, bis zu der Variablen innerhalb eines Knotens als Alternative zur bestbewerteten untersucht werden, auf `w`. Ist beispielsweise $w = 5$, so werden innerhalb eines Knotens höchstens die besten fünf Variablen weiterverfolgt.[10]

- `-randomnoise`: Sorgt dafür, dass *Solver* die von der Bewertungsfunktion $\mathscr{E}$ gelieferten Werte mit einem kleinen zufälligen Rauschen überlagert. Die Idee ist, die Gefahr von „blind spots“ in der Suche zu reduzieren, freilich unter gewissen Verlusten in der Bewertungsgenauigkeit. Obwohl eine exakte Begründung schwer fällt, werden nach subjektivem Empfinden des Autors damit Lösungen im Mittel schneller bzw. zuverlässiger gefunden. Daher wird die Verwendung dieser Option empfohlen.

- `-t <timeout>`: Bricht die Suche ab, wenn nach `timeout` Sekunden noch keine Lösung gefunden wurde.

- `-timeadaption`: Nur sinnvoll, wenn auch `-t` gesetzt ist. Das Programm versucht dann, durch dynamische Vergröberung der Suchgenauigkeit innerhalb der vorgegebenen Zeit möglichst gleichmäßig alle Regionen des Suchbaumes zu betrachten, statt eventuell nur einen kleinen Teilast sehr detailliert zu untersuchen. Die Verwendung dieser Option ist empfohlen.

- `-evalhashsize <s>`: Setzt die maximale Größe der Hashtabelle für die heuristische Isomorphieerkennung nach Kapitel 3.7 auf `s` Megabyte.

- `-permhashsize <s>`: Wie zuvor, aber für eine andere Hashtabelle: Neben derjenigen für die heuristische Isomorphieerkennung führt *Solver* auch noch eine Tabelle mit Hashwerten zu den bisher im Rahmen von Algorithmus 3.3 untersuchten Vektoren $x^{(i)}$. Dies ist sinnvoll, da derselbe Vektor durch unterschiedliche Reihenfolgen bei der Variableninkrementierung erreicht werden kann. Im Gegensatz zur heuristischen Variante ist in diesem Fall eine Suchbaumabschneidung vollkommen sicher.[11] Natürlich würde hier insbesondere auch die heuristische Isomorphieerkennung anschlagen, allerdings müssten dafür zunächst die Werte von $\mathscr{E}$ für alle möglichen Vektoren, die aus $x^{(i)}$ durch Inkrementierung einer Komponente hervorgehen, bestimmt werden. In der zweiten Variante dagegen kann die Wiederholung schon bei der Wahl im Vorgängerknoten bemerkt und damit wertvolle Rechenzeit gespart werden.

- `-noevalhash`: Schaltet die Nutzung des Hashes zur heuristischen Isomorphieerkennung komplett ab. In Kombination mit einem extrem großen Wert für den Schalter

[10] Weniger sind möglich aufgrund des Einflusses des Parameters a im Backtracking.
[11] Denkbar sind selbstverständlich immer noch Fehler durch Hashkollisionen.

`-a` (etwa `-a 1.0e100`) kann hiermit erzwungen werden, dass das System ohne jede heuristische Reduktion auf Lösungen durchsucht wird. Ein Scheitern der Suche ist in diesem Fall gleichbedeutend mit der Nichtexistenz einer Lösung.

- `-u <u>`: Sorgt dafür, dass alle u Sekunden eine Ausgabe stattfindet.
- `-o <solfile>`: Schreibt im Erfolgsfall die Lösung in die Datei `solfile`.

Kapitel 7.

Ergebnisse

In diesem Kapitel sollen die mit den Programmen *Heurico* und *Solver* erzielten Ergebnisse vorgestellt werden. Mit ersterem wurden nur Suchen nach linearen Codes über echten Kettenringen (d. h. mit $m > 1$) durchgeführt, da der speziellere Fall für Körper schon in [Zwa07] behandelt wurde. *Solver* kam dagegen zur Konstruktion linearer Codes mit vorgeschriebener Automorphismengruppe, sowohl über Körpern als auch über allgemeinen Kettenringen, zum Einsatz.

7.1. Lineare Codes mit vorgeschriebener Automorphismengruppe über Körpern

Die Kramer-Mesner-Matrizen stammen in diesem Fall aus der Datenbank von A. Kohnert, der auch die Suche mit *Solver* zu einem Großteil durchführte. In Anhang C geben wir Tabellen zu den Stellen, bei denen die untere Schranke für lineare (n, q^k, d)-Codes über $\mathbb{F}_q$ in [Gra] verbessert werden konnte.[1] Insgesamt handelt es sich um 497 Neuerungen, 38 dieser Codes treffen die in [Gra] angegebene obere Schranke und sind daher optimal. Ein kleiner Teil der Daten wurde bereits in [KZ09] veröffentlicht. Details zu den Codes, insbesondere zur verwendeten Automorphismengruppe, können online abgefragt werden [Koh].

7.2. Lineare Codes über Kettenringen

7.2.1. Tabellen auf der CD

Die Tabellen zu den mit *Heurico* und *Solver* gefundenen Codes wurden aufgrund des beträchtlichen Umfanges auf die beigefügte CD ausgelagert.[2] Die Betrachtung ist mit jedem gewöhnlichen Internetbrowser möglich. Zum Einstieg muss lediglich die Datei `index.html` im Verzeichnis `tables` mit dem Browser geöffnet werden, die weitere Bedienung ist weitestgehend selbsterklärend. Je Kettenring R existiert eine Tabelle, in der

[1] Wir zählen hier auch die Verbesserungen mit auf, die man durch Standardoperationen wie Punktieren oder den Paritycheck (bei binären Codes) erhält. Bei optimalen Codes ist der Eintrag für die Minimaldistanz fett gedruckt.

[2] Die Betrachtung ist auch online möglich, siehe [Zwa].

für jede Kombination aus Länge und Größe der Codes die beste gefundene homogene Minimaldistanz aufgelistet ist. Weitere Übersichten zeigen die besten gefundenen Codes einer festen Kardinalität zu verschiedenen Umrissen. Ferner sind Tabellen angefügt, in denen die Minimaldistanz der Graybilder mit den besten bekannten linearen Codes derselben Größe und Länge über dem entsprechenden Körper $\mathbb{F}_q$ verglichen wird. Diese sind aufgeschlüsselt nach q und der Kardinalität der Codes. Der Vergleich wird erleichtert durch ein innerhalb der Tabellen erläutertes Farbschema.

7.2.2. Zahlen zur Datenbank

Der Großteil der Daten, nämlich über 90.000 Codes, wurde mit dem Programm *Heurico* (vgl. Abschnitt 6.1) konstruiert. Anschließend konnten die so gewonnenen Ergebnisse mit *Solver* (vgl. Abschnitt 6.2) noch einmal an etwa 2.300 Stellen verbessert werden. Insgesamt wurden damit rund 93.000 lineare Codes für 24 verschiedene Kettenringe generiert.

Gilt $\mathrm{char}(R) = p$, so ist das Graybild eines R-linearen Codes nach Lemma 2.7 auch linear über dem zugrundeliegenden Restklassenkörper $\mathbb{F}_q$. In diesem Fall kann aus einer oberen Schranke für die Hammingdistanz linearer Codes über $\mathbb{F}_q$ mit Hilfe von Satz 2.6 eine obere Schranke für die homogene Minimaldistanz der Codes über R abgeleitet werden. Konkret gilt für die homogene Minimaldistanz d_{hom} eines linearen Codes C der Länge n über R mit $|C| = q^k$:

$$q^{m-2} d_{\mathrm{hom}} \leq u(q^{m-1}n, k, q),$$

wobei $u(n', k, q)$ eine obere Schranke für die Hammingdistanz linearer Codes der Länge n' und Dimension k über $\mathbb{F}_q$ ist. Auf diese Weise kann die Optimalität für über 1.200 Codes in der Datenbank nachgewiesen werden.[3]

7.3. Interessante Funde

Lineare Codes über einem Kettenring, deren Graybild bessere Minimaldistanz aufweist als alle bisher bekannten linearen Codes mit denselben Parametern über dem Grundkörper $\mathbb{F}_q$, wollen wir kurz als *BTKL-Codes* („better-than-known-linear“) bezeichnen. Ist sogar beweisbar, dass entsprechende $\mathbb{F}_q$-lineare Codes nicht existieren können, sprechen wir von *BTL*-Codes („better-than-linear“). In der Datenbank auf der CD befinden sich

[3] Diese Codes entsprechen den grün hinterlegten Einträgen in den ringspezifischen Tabellen auf der CD.

BTL- und BTKL-Codes zu folgenden Parametern:

Ring	Umriss	n	d_{hom}	Typ	Quelle
$\mathbb{Z}_4$	$(2,2,2)$	7	6	BTL	verkürzter $\mathcal{K}(4)$
	$(2,2,2,1)$	29	28	BTL	[KZ]
	$(2,2,2,2)$	8	6	BTL	$\mathcal{K}(4)$
		30	28	BTL	[ARC02]
		57	56	BTL	[KZ10]
	$(2,2,2,2,1)$	79	76	BTKL	
		85	81	BTKL	
		87	84	BTKL	
		90	88	BTKL	
	$(2,2,2,2,2)$	30	26	BTKL	Punktieren von verkürztem $\mathcal{K}(6)$
		31	28	BTL	verkürzter $\mathcal{K}(6)$
	$(2,2,2,2,2,1)$	27	22	BTKL	
		31	26	BTKL	Untercode von punktiertem $\mathcal{K}(6)$
		32	28	BTL	Untercode von $\mathcal{K}(6)$
	$(2,2,2,2,2,2)$	24	18	BTKL	
		31	26	BTL	punktierter $\mathcal{K}(6)$
		32	28	BTL	$\mathcal{K}(6)$
		33	28	BTKL	beliebiges Verlängern von $\mathcal{K}(6)$
		41	34	BTKL	
		49	42	BTKL	
	$(2,2,2,2,2,2,1)$	34	28	BTKL	
		42	34	BTKL	[ARC02]
GR$(16,4)$	$(2,2,2)$	21	60	BTKL	Hyperoval in PHG$(2,\text{GR}(16,4))$
	$(2,2,2,2)$	63	177	BTKL	

Mit $\mathcal{K}(u)$ ist dabei eine $\mathbb{Z}_4$-lineare Darstellung des binären Kerdock-Codes mit den Parametern $(2^u, 2^{2u}, 2^{u-1} - 2^{\frac{u-2}{2}})$ gemeint, wie sie in [Nec91] bzw. [HKC+94] hergeleitet wurde. Das *Hyperoval*[4], aus dessen Punktmenge die Spalten einer Generatormatrix des linearen $(21, 16^3, 60)$-Codes über GR$(16,4)$ hervorgehen, wurde in [HW99] entdeckt.[5] In [HL05] wurde die Existenz von Hyperovalen dann für alle Galoisringe vom Typ GR$(4^r, 4)$ nachgewiesen. Der lineare $(63, 16^4, 177)$-Code über GR$(16,4)$ ergibt sich durch geeignetes Punktieren[6] aus dem Code $\mathcal{K}_{\text{GR}(16,4)}(3)$, der in [KN97] definiert wird. Dort erfolgt auch eine Bestimmung des zugehörigen homogenen Gewichtszählers.[7] Weitere Parametersätze in der obigen Tabelle, bei denen sich eine Quellenangabe findet, wurden bereits an entsprechender Stelle veröffentlicht. Für alle anderen ist die Existenz entsprechender Codes unseres Wissens ein neues Resultat.

[4] Unter einem Hyperoval in PHG$(2,R)$ versteht man einen $(q^2+q+1, 2)-Arc$.

[5] vgl. auch [HHL00]

[6] Bei Punktieren an einer beliebigen Stelle kann die homogene Minimaldistanz auf 176 sinken.

[7] Das Graybild von $\mathcal{K}_{\text{GR}(16,4)}(3)$ über $\mathbb{F}_4$ ist der verallgemeinerte Kerdock-Code $\mathcal{K}_4(4)$ [KN94]. Die Gewichtszähler der verallgemeinerten Kerdock-Codes bezüglich des Hamminggewichts wurden in [KN96] berechnet.

Zu dem $\mathbb{Z}_4$-linearen Code mit Umriss $\lambda = (2, 2, 2, 1)$, Länge 29 und Minimaldistanz $d_{\text{hom}} = 28$ konnte eine interessante geometrische Interpretation gefunden werden, die in [KZ] veröffentlicht wurde. Sein Graybild verbessert außerdem die in [LRS] angegebene untere Schranke für die Kardinalität binärer Blockcodes mit Länge 58 und minimaler Hammingdistanz 28 von 124 [ZL87] auf 128. Entsprechendes gilt für den $\mathbb{Z}_4$-linearen Code mit Umriss $\lambda = (2, 2, 2, 2, 2, 2)$, hier wurde die untere Schranke für die Größe binärer Blockcodes der Länge 48 und minimaler Hammingdistanz 18 von 2560 [Lit98] auf 4096 verbessert.

Der angegebene lineare $(57, 4^4, 56)$-Code über $\mathbb{Z}_4$ entpuppte sich als das erste Glied einer unendlichen Serie sehr guter $\mathbb{Z}_4$-linearer Codes, die in [KZ10] beschrieben wird. Auch für den zweiten Vertreter dieser Serie, einen linearen $(994, 4^6, 992)$-Code, konnte inzwischen nachgewiesen werden, dass es sich um einen BTL-Code handelt [WZ10].

7.4. Fazit

Der in [Zwa07] verwendete Ansatz zur heuristischen Suche nach linearen Codes über endlichen Körpern konnte deutlich erweitert werden: Mit den hier beschriebenen Algorithmen ist es möglich, auch allgemeinere Diophantische Ungleichungssysteme auf heuristische Weise zu lösen. Die Leistungsfähigkeit dieser Algorithmen demonstrieren die Programme *Heurico* und *Solver*, mit deren Hilfe eine umfangreiche Datenbank für lineare Codes über endlichen Kettenringen aufgebaut wurde. Zuvor gab es unseres Wissens lediglich die auf $\mathbb{Z}_4$-lineare Codes spezialisierte Datenbank von Tsvetan Asamov [Asa], deren Einträge von den hier konstruierten Codes an sehr vielen Stellen deutlich verbessert werden.

Ferner gelang es, weitere lineare Codes über $\mathbb{Z}_4$ zu finden, deren Graybild die bekannten unteren bzw. oberen Schranken für die Minimaldistanz entsprechender binärer linearer Codes übertrifft; in einem Fall konnte sogar eine sehr gute unendliche Serie abgeleitet werden. Für andere Kettenringe förderte die Suche nur ein neues Beispiel zu Tage, nämlich den linearen $(63, 16^4, 177)$-Code über $\mathrm{GR}(16, 4)$. Unterstellt man, dass die Datenbank nicht zu viele Lücken enthält, muss man schließen, dass derartige Codes „selten“ sind. Das macht sie umgekehrt natürlich auch zu sehr interessanten Objekten.

Anhang A.

Tabellen zu Beispiel 4.2

Additionstabelle für den Kettenring $\mathbb{F}_4[X,\sigma]/(X^2)$:

+	0	1	a	(a+1)	X	X+1	X+a	X+(a+1)	aX	aX+1	aX+a	aX+(a+1)	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)
0	0	1	a	(a+1)	X	X+1	X+a	X+(a+1)	aX	aX+1	aX+a	aX+(a+1)	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)
1	1	0	(a+1)	a	X+1	X	X+(a+1)	X+a	aX+1	aX	aX+(a+1)	aX+a	(a+1)X+1	(a+1)X	(a+1)X+(a+1)	(a+1)X+a
a	a	(a+1)	0	1	X+a	X+(a+1)	X	X+1	aX+a	aX+(a+1)	aX	aX+1	(a+1)X+a	(a+1)X+(a+1)	(a+1)X	(a+1)X+1
(a+1)	(a+1)	a	1	0	X+(a+1)	X+a	X+1	X	aX+(a+1)	aX+a	aX+1	aX	(a+1)X+(a+1)	(a+1)X+a	(a+1)X+1	(a+1)X
X	X	X+1	X+a	X+(a+1)	0	1	a	(a+1)	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)	aX	aX+1	aX+a	aX+(a+1)
X+1	X+1	X	X+(a+1)	X+a	1	0	(a+1)	a	(a+1)X+1	(a+1)X	(a+1)X+(a+1)	(a+1)X+a	aX+1	aX	aX+(a+1)	aX+a
X+a	X+a	X+(a+1)	X	X+1	a	(a+1)	0	1	(a+1)X+a	(a+1)X+(a+1)	(a+1)X	(a+1)X+1	aX+a	aX+(a+1)	aX	aX+1
X+(a+1)	X+(a+1)	X+a	X+1	X	(a+1)	a	1	0	(a+1)X+(a+1)	(a+1)X+a	(a+1)X+1	(a+1)X	aX+(a+1)	aX+a	aX+1	aX
aX	aX	aX+1	aX+a	aX+(a+1)	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)	0	1	a	(a+1)	X	X+1	X+a	X+(a+1)
aX+1	aX+1	aX	aX+(a+1)	aX+a	(a+1)X+1	(a+1)X	(a+1)X+(a+1)	(a+1)X+a	1	0	(a+1)	a	X+1	X	X+(a+1)	X+a
aX+a	aX+a	aX+(a+1)	aX	aX+1	(a+1)X+a	(a+1)X+(a+1)	(a+1)X	(a+1)X+1	a	(a+1)	0	1	X+a	X+(a+1)	X	X+1
aX+(a+1)	aX+(a+1)	aX+a	aX+1	aX	(a+1)X+(a+1)	(a+1)X+a	(a+1)X+1	(a+1)X	(a+1)	a	1	0	X+(a+1)	X+a	X+1	X
(a+1)X	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)	aX	aX+1	aX+a	aX+(a+1)	X	X+1	X+a	X+(a+1)	0	1	a	(a+1)
(a+1)X+1	(a+1)X+1	(a+1)X	(a+1)X+(a+1)	(a+1)X+a	aX+1	aX	aX+(a+1)	aX+a	X+1	X	X+(a+1)	X+a	1	0	(a+1)	a
(a+1)X+a	(a+1)X+a	(a+1)X+(a+1)	(a+1)X	(a+1)X+1	aX+a	aX+(a+1)	aX	aX+1	X+a	X+(a+1)	X	X+1	a	(a+1)	0	1
(a+1)X+(a+1)	(a+1)X+(a+1)	(a+1)X+a	(a+1)X+1	(a+1)X	aX+(a+1)	aX+a	aX+1	aX	X+(a+1)	X+a	X+1	X	(a+1)	a	1	0

Multiplikationstabelle für den Kettenring $\mathbb{F}_4[X,\sigma]/(X^2)$:

·	0	1	a	(a+1)	X	X+1	X+a	X+(a+1)	aX	aX+1	aX+a	aX+(a+1)	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	a	(a+1)	X	X+1	X+a	X+(a+1)	aX	aX+1	aX+a	aX+(a+1)	(a+1)X	(a+1)X+1	(a+1)X+a	(a+1)X+(a+1)
a	0	a	(a+1)	1	aX	aX+a	aX+(a+1)	aX+1	(a+1)X	(a+1)X+a	(a+1)X+(a+1)	(a+1)X+1	X	X+a	X+(a+1)	X+1
(a+1)	0	(a+1)	1	a	(a+1)X	(a+1)X+(a+1)	(a+1)X+1	(a+1)X+a	X	X+(a+1)	X+1	X+a	aX	aX+(a+1)	aX+1	aX+a
X	0	X	(a+1)X	aX	0	X	(a+1)X	aX	0	X	(a+1)X	aX	0	X	(a+1)X	aX
X+1	0	X+1	(a+1)X+a	aX+(a+1)	X	1	aX+a	(a+1)X+(a+1)	aX	(a+1)X+1	X+a	(a+1)	(a+1)X	aX+1	a	X+(a+1)
X+a	0	X+a	(a+1)X+(a+1)	aX+1	aX	(a+1)X+a	X+(a+1)	1	(a+1)X	aX+a	(a+1)	X+1	X	a	aX+(a+1)	(a+1)X+1
X+(a+1)	0	X+(a+1)	(a+1)X+1	aX+a	(a+1)X	aX+(a+1)	1	X+a	X	(a+1)	aX+1	(a+1)X+a	aX	(a+1)X+(a+1)	X+1	a
aX	0	aX	X	(a+1)X	0	aX	X	(a+1)X	0	aX	X	(a+1)X	0	aX	X	(a+1)X
aX+1	0	aX+1	X+a	(a+1)X+(a+1)	X	(a+1)X+1	a	aX+(a+1)	aX	1	(a+1)X+a	X+(a+1)	(a+1)X	X+1	aX+a	(a+1)
aX+a	0	aX+a	X+(a+1)	(a+1)X+1	aX	a	(a+1)X+(a+1)	X+1	(a+1)X	X+a	aX+(a+1)	1	X	(a+1)X+a	(a+1)	aX+1
aX+(a+1)	0	aX+(a+1)	X+1	(a+1)X+a	(a+1)X	X+(a+1)	aX+1	a	X	(a+1)X+(a+1)	1	aX+a	aX	(a+1)	(a+1)X+1	X+a
(a+1)X	0	(a+1)X	aX	X	0	(a+1)X	aX	X	0	(a+1)X	aX	X	0	(a+1)X	aX	X
(a+1)X+1	0	(a+1)X+1	aX+a	X+(a+1)	X	aX+1	(a+1)X+a	(a+1)	aX	X+1	a	(a+1)X+(a+1)	(a+1)X	1	X+a	aX+(a+1)
(a+1)X+a	0	(a+1)X+a	aX+(a+1)	X+1	aX	X+a	(a+1)	(a+1)X+1	(a+1)X	a	X+(a+1)	aX+1	X	aX+a	(a+1)X+(a+1)	1
(a+1)X+(a+1)	0	(a+1)X+(a+1)	aX+1	X+a	(a+1)X	(a+1)	X+1	aX+a	X	aX+(a+1)	(a+1)X+1	a	aX	X+(a+1)	1	(a+1)X+a

Verknüpfungstafel zum Ringautomorphismus ψ:

r	0	1	a	$a+1$	X	$(X+1)$	$X+a$	$X+a+1$	aX	$aX+1$	$aX+a$	$aX+a+1$	$(a+1)X$	$(a+1)X+1$	$(a+1)X+a$	$(a+1)X+a+1$
$\psi(r)$	0	1	$aX+a+1$	$aX+a$	$(a+1)X$	$(a+1)X+1$	$X+a+1$	$X+a$	aX	$aX+1$	$a+1$	a	X	$X+1$	$(a+1)X+a+1$	$(a+1)X+a$

Die Matrix $\mathcal{M}^\chi$ (bei Reduktion der Zeilen auf einen Vertreter pro Bahn unter der Operation von G) ist der innere Teil der folgenden Tabelle:

	1	1	1	1	1	1	aX	$(a+1)X$	1	1	1	1	1	1	1	1	1	1	0	X	X	X	X	0	X
	0	1	$a+1$	$X+a+1$	aX	$aX+1$	1	1	a	$X+a$	$aX+a$	$(a+1)X+a$	X	$X+1$	$aX+a+1$	$(a+1)X$	$(a+1)X+1$	$(a+1)X+a+1$	1	1	0	X	$(a+1)X$	X	aX
$(1,0)$	X_2	X_2	X_2	X_2	X_2	X_2	X_1	X_1	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_0	X_1	X_1	X_1	X_1	X_0	X_1
$(1,1)$	X_2	X_0	X_2	X_2	X_2	X_1	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_1	X_2	X_2	X_1	X_2	X_2	X_2	X_1	X_0	X_1	X_1	X_1
$(1,a+1)$	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_0	X_1	X_1	X_1	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_1	X_1	X_1	X_1	X_0
$(X,0)$	X_1	X_1	X_1	X_1	X_1	X_1	X_0	X_0	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_0	X_0	X_0	X_0	X_0	X_0	X_0
(X,aX)	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_0	X_0	X_0	X_0	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_1	X_0	X_0	X_0	X_0	X_0

Anhang B.

Polynome zu Beispiel 4.3

Für die Berechnung der Polynome $p_i(\xi,\eta)$ bestimmen wir zunächst die $d^i_{o,m}$. Es gilt:

$$\begin{aligned}
&d^0_{8,26}=1,\ d^0_{8,22}=1,\ d^0_{4,12}=2,\ d^0_{1,4}=1\\
&d^1_{8,26}=1,\ d^1_{8,22}=1,\ d^1_{4,12}=2,\ d^1_{1,4}=1\\
&d^2_{8,24}=2,\ d^2_{4,16}=1,\ d^2_{4,12}=1,\ d^2_{1,0}=1\\
&d^3_{8,24}=2,\ d^3_{4,16}=1,\ d^3_{4,0}=1,\ d^3_{1,0}=1\\
&d^4_{8,32}=2,\ d^4_{4,0}=2,\ d^4_{1,0}=1.
\end{aligned}$$

Nach Definition lauten die Polynome dann:

$$\begin{aligned}
p_0(\xi,\eta) = {} & (\xi^{16}\eta^{52}+\xi^8\eta^{26}+1)\cdot(\xi^{16}\eta^{44}+\xi^8\eta^{22}+1)\cdot\\
& \cdot(6\xi^{20}\eta^{60}+5\xi^{16}\eta^{48}+4\xi^{12}\eta^{36}+3\xi^8\eta^{24}+2\xi^4\eta^{12}+1)\cdot\\
& \cdot(\xi^{20}\eta^{80}+\xi^{19}\eta^{76}+\xi^{18}\eta^{72}+\xi^{17}\eta^{68}+\xi^{16}\eta^{64}+\xi^{15}\eta^{60}+\xi^{14}\eta^{56}+\xi^{13}\eta^{52}+\\
& +\xi^{12}\eta^{48}+\xi^{11}\eta^{44}+\xi^{10}\eta^{40}+\xi^9\eta^{36}+\xi^8\eta^{32}+\xi^7\eta^{28}+\xi^6\eta^{24}+\xi^5\eta^{20}+\xi^4\eta^{16}+\\
& +\xi^3\eta^{12}+\xi^2\eta^8+\xi^1\eta^4+1).\\
p_1(\xi,\eta) = {} & p_0(\xi,\eta).\\
p_2(\xi,\eta) = {} & (3\xi^{16}\eta^{48}+2\xi^8\eta^{24}+1)\cdot(\xi^{20}\eta^{80}+\xi^{16}\eta^{64}+\xi^{12}\eta^{48}+\xi^8\eta^{32}+\xi^4\eta^{16}+1)\cdot\\
& \cdot(\xi^{20}\eta^{60}+\xi^{16}\eta^{48}+\xi^{12}\eta^{36}+\xi^8\eta^{24}+\xi^4\eta^{12}+1)\cdot(\xi^{20}+\xi^{19}+\xi^{18}+\xi^{17}+\xi^{16}+\\
& +\xi^{15}+\xi^{14}+\xi^{13}+\xi^{12}+\xi^{11}+\xi^{10}+\xi^9+\xi^8+\xi^7+\xi^6+\xi^5+\xi^4+\xi^3+\xi^2+\\
& +\xi^1+1).\\
p_3(\xi,\eta) = {} & (3\xi^{16}\eta^{48}+2\xi^8\eta^{24}+1)\cdot(\xi^{20}\eta^{80}+\xi^{16}\eta^{64}+\xi^{12}\eta^{48}+\xi^8\eta^{32}+\xi^4\eta^{16}+1)\cdot\\
& \cdot(\xi^{20}+\xi^{16}+\xi^{12}+\xi^8+\xi^4+1)\cdot(\xi^{20}+\xi^{19}+\xi^{18}+\xi^{17}+\xi^{16}+\xi^{15}+\xi^{14}+\xi^{13}+\\
& +\xi^{12}+\xi^{11}+\xi^{10}+\xi^9+\xi^8+\xi^7+\xi^6+\xi^5+\xi^4+\xi^3+\xi^2+\xi+1).\\
p_4(\xi,\eta) = {} & (3\xi^{16}\eta^{64}+2\xi^8\eta^{32}+1)\cdot(6\xi^{20}+5\xi^{16}+4\xi^{12}+3\xi^8+2\xi^4+1)\cdot\\
& \cdot(\xi^{20}+\xi^{19}+\xi^{18}+\xi^{17}+\xi^{16}+\xi^{15}+\xi^{14}+\xi^{13}+\xi^{12}+\xi^{11}+\xi^{10}+\xi^9+\xi^8+\xi^7+\\
& +\xi^6+\xi^5+\xi^4+\xi^3+\xi^2+\xi^1+1).
\end{aligned}$$

Ausmultipliziert ergibt das, unter Verwendung der beschriebenen Reduktionen:

$$
\begin{aligned}
p_0(\xi,\eta) \equiv\ & \xi^{20}(44\eta^{60}+4\eta^{58}+2\eta^{56})+\xi^{19}(26\eta^{60}+3\eta^{58}+\eta^{56})+\xi^{18}(15\eta^{60}+5\eta^{58}+\\
&+6\eta^{56}+3\eta^{54}+\eta^{52})+\xi^{17}(7\eta^{60}+3\eta^{58}+5\eta^{56}+5\eta^{54}+6\eta^{52}+3\eta^{50}+\eta^{48})+\\
&+\xi^{16}(3\eta^{60}+\eta^{58}+3\eta^{56}+3\eta^{54}+5\eta^{52}+5\eta^{50}+6\eta^{48}+3\eta^{46}+\eta^{44})+\\
&+\xi^{15}(\eta^{60}+2\eta^{56}+\eta^{54}+3\eta^{52}+3\eta^{50}+4\eta^{48}+2\eta^{46})+\xi^{14}(\eta^{56}+2\eta^{52}+\\
&+\eta^{50}+3\eta^{48}+3\eta^{46}+4\eta^{44}+2\eta^{42})+\xi^{13}(\eta^{52}+2\eta^{48}+\eta^{46}+3\eta^{44}+3\eta^{42}+\\
&+4\eta^{40}+2\eta^{38})+\xi^{12}(\eta^{48}+2\eta^{44}+\eta^{42}+3\eta^{40}+3\eta^{38}+4\eta^{36}+2\eta^{34})+\\
&+\xi^{11}(\eta^{44}+2\eta^{40}+\eta^{38}+3\eta^{36}+\eta^{34})+\xi^{10}(\eta^{40}+2\eta^{36}+\eta^{34}+3\eta^{32}+\eta^{30})+\\
&+\xi^{9}(\eta^{36}+2\eta^{32}+\eta^{30}+3\eta^{28}+\eta^{26})+\xi^{8}(\eta^{32}+2\eta^{28}+\eta^{26}+3\eta^{24}+\eta^{22})+\\
&+\xi^{7}(\eta^{28}+2\eta^{24})+\xi^{6}(\eta^{24}+2\eta^{20})+\xi^{5}(\eta^{20}+2\eta^{16})+\xi^{4}(\eta^{16}+2\eta^{12})+\\
&+\xi^{3}\eta^{12}+\xi^{2}\eta^{8}+\xi\eta^{4}+1.\\
p_2(\xi,\eta) \equiv\ & \xi^{20}(22\eta^{60}+3\eta^{56}+3\eta^{52}+7\eta^{48}+\eta^{44}+3\eta^{40}+3\eta^{36}+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\\
&+\eta^{12}+1)+\xi^{19}(2\eta^{60}+3\eta^{56}+3\eta^{52}+7\eta^{48}+\eta^{44}+3\eta^{40}+3\eta^{36}+\eta^{32}+\eta^{28}+\\
&+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{18}(2\eta^{60}+3\eta^{56}+3\eta^{52}+7\eta^{48}+\eta^{44}+3\eta^{40}+3\eta^{36}+\\
&+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{17}(2\eta^{60}+3\eta^{56}+3\eta^{52}+7\eta^{48}+\eta^{44}+\\
&+3\eta^{40}+3\eta^{36}+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{16}(2\eta^{60}+3\eta^{56}+3\eta^{52}+\\
&+7\eta^{48}+\eta^{44}+3\eta^{40}+3\eta^{36}+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{15}(\eta^{48}+\\
&+\eta^{44}+3\eta^{40}+3\eta^{36}+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{14}(\eta^{48}+\eta^{44}+\\
&+3\eta^{40}+3\eta^{36}+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{13}(\eta^{48}+\eta^{44}+3\eta^{40}+\\
&+3\eta^{36}+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{12}(\eta^{48}+\eta^{44}+3\eta^{40}+3\eta^{36}+\\
&+\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{11}(\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\\
&+\xi^{10}(\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{9}(\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\\
&+\xi^{8}(\eta^{32}+\eta^{28}+3\eta^{24}+\eta^{16}+\eta^{12}+1)+\xi^{7}(\eta^{16}+\eta^{12}+1)+\xi^{6}(\eta^{16}+\eta^{12}+1)+\\
&+\xi^{5}(\eta^{16}+\eta^{12}+1)+\xi^{4}(\eta^{16}+\eta^{12}+1)+\xi^{3}+\xi^{2}+\xi^{1}+1.\\
p_3(\xi,\eta) \equiv\ & \xi^{20}(8\eta^{60}+4\eta^{56}+9\eta^{48}+6\eta^{40}+4\eta^{32}+8\eta^{24}+5\eta^{16}+6)+\xi^{19}(\eta^{60}+2\eta^{56}+\\
&+5\eta^{48}+4\eta^{40}+3\eta^{32}+6\eta^{24}+4\eta^{16}+5)+\xi^{18}(\eta^{60}+2\eta^{56}+5\eta^{48}+4\eta^{40}+\\
&+3\eta^{32}+6\eta^{24}+4\eta^{16}+5)+\xi^{17}(\eta^{60}+2\eta^{56}+5\eta^{48}+4\eta^{40}+3\eta^{32}+6\eta^{24}+\\
&+4\eta^{16}+5)+\xi^{16}(\eta^{60}+2\eta^{56}+5\eta^{48}+4\eta^{40}+3\eta^{32}+6\eta^{24}+4\eta^{16}+5)+\\
&+\xi^{15}(\eta^{48}+2\eta^{40}+2\eta^{32}+4\eta^{24}+3\eta^{16}+4)+\xi^{14}(\eta^{48}+2\eta^{40}+2\eta^{32}+\\
&+4\eta^{24}+3\eta^{16}+4)+\xi^{13}(\eta^{48}+2\eta^{40}+2\eta^{32}+4\eta^{24}+3\eta^{16}+4)+\xi^{12}(\eta^{48}+\\
&+2\eta^{40}+2\eta^{32}+4\eta^{24}+3\eta^{16}+4)+\xi^{11}(\eta^{32}+2\eta^{24}+2\eta^{16}+3)+\xi^{10}(\eta^{32}+\\
&+2\eta^{24}+2\eta^{16}+3)+\xi^{9}(\eta^{32}+2\eta^{24}+2\eta^{16}+3)+\xi^{8}(\eta^{32}+2\eta^{24}+2\eta^{16}+3)+\\
&+\xi^{7}(\eta^{16}+2)+\xi^{6}(\eta^{16}+2)+\xi^{5}(\eta^{16}+2)+\xi^{4}(\eta^{16}+2)+\xi^{3}+\xi^{2}+\xi^{1}+1.\\
p_4(\xi,\eta) \equiv\ & \xi^{20}(9\eta^{60}+20\eta^{32}+21)+\xi^{19}(3\eta^{60}+12\eta^{32}+15)+\xi^{18}(3\eta^{60}+12\eta^{32}+15)+\\
&+\xi^{17}(3\eta^{60}+12\eta^{32}+15)+\xi^{16}(3\eta^{60}+12\eta^{32}+15)+\xi^{15}(6\eta^{32}+10)+\\
&+\xi^{14}(6\eta^{32}+10)+\xi^{13}(6\eta^{32}+10)+\xi^{12}(6\eta^{32}+10)+\xi^{11}(2\eta^{32}+6)+\xi^{10}(2\eta^{32}+6)+\\
&+\xi^{9}(2\eta^{32}+6)+\xi^{8}(2\eta^{32}+6)+3\xi^{7}+3\xi^{6}+3\xi^{5}+3\xi^{4}+\xi^{3}+\xi^{2}+\xi^{1}+1.
\end{aligned}
$$

Anhang C.

Neue lineare Codes mit der Kramer-Mesner-Methode über Körpern

C.1. Ergebnisse[1] über $\mathbb{F}_2$

n/k	9	11	12	13	14	15	16
40		**15**		13			
41		**16**		**14**			
42	**17**						
43	**18**						
50	**21**				17		
51	**22**				**18**		
53	**23**						
54	**24**						
58	**25**		23				
59	**26**		**24**	23			
60				**24**			
67			27				
68			**28**				
76				31			
77				**32**			
79		33					
80		34					
82		35					
83		**36**					
93							35
94							36
95		41					
96		42					
98			41				
99			42				
101			43				

n/k	9	11	12	13	14	15	16
102			44				
106	**49**						
107	**50**						
111					47		
112					48		
142	67						
143	**68**						
150			67	65			
151			68	66			
154		71					
155		**72**					
160							65
161							66
167				73			
168				74			
170		79		75			
171		**80**		76			
174						75	
175						76	
252					113		
253					114		

Tabelle C.1.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_2$.

[1] Fett gedruckte Einträge sind optimal.

C.2. Ergebnisse[2] über $\mathbb{F}_3$

n/k	7	8	9	10	11	12
26			**12**			
36	**20**					
37	**21**					
38		20				
39		**21**				
47	**27**					
48		**27**				
52		29				
53		30	28			
54			29			
55			30	28		
56				29		
57		32		30		
58		33				
62			34	33		
63			35		32	
64			36		33	
65				34		
66		38		35		
67		39		36		
74		43				
75		44				
76		45				
98				55		
99				56		52
100				57		53
101						54
102		61				
103		62				
104		63				
105				60		
107		64				
108		65				
109		66				
111			65			
112			66			
113						61

n/k	7	8	9	10	11	12
114						62
115						63
116		70				
117		71				
118		72				
121		73				
122		74			69	
123		75				
124			73			
125			74			
126			75			
128			76			
129			77			
130			78			
133						73
134						74
135						75
137						76
138			82			77
139			83			78
140			84			
143			85			
144			86		82	
145			87		83	
146					84	
150					85	
151					86	
152			91		87	
153			92			85
154			93		88	86
155					89	87
156					90	
159					91	
160	102				92	
161					93	
164					94	
165					95	
166					96	
167						94

[2]Fett gedruckte Einträge sind optimal.

n/k	7	8	9	10	11	12
168			102			95
169					97	96
170		106			98	
171		107			99	
172		108				
173					100	
174					101	
175					102	
176		109		105		100
177		110				101
178		111			103	102
179					104	
180		112			105	
181		113				
182		114			106	
183					107	
184				109	108	
185		115		110		
186		116		111		
187		117				
188						108
190			116		111	
191			117			
193				115		
194			118	116	114	
195			119	117		
196			120			
199				119		
200			123	120		
202				121		
203			124	122		
204			125	123		
205			126			
206		129				
208			127			
209			128			
210			129			
212			130			123
213			131			

n/k	7	8	9	10	11	12
214			132			
218		137	134			
219		138	135			
220			136			
221			137			129
222			138			
223			139			
224			140			
225			141			

Tabelle C.2.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_3$.

C.3. Ergebnisse[3] über $\mathbb{F}_4$

n/k	6	7	8	9
25		**14**		
29	**18**			
34			19	
35		21	20	
36	23	22		
37	**24**		21	
39				21
43				24
46		29		
47		30		
48		31		
49		**32**		
80				49
81				50
90			59	
91			60	
94		64		
96			63	
97			64	
99		67		
100		68		
102				65
103				66
104	73			67
105	74			68
106	75			
107	76	73		
108		74		69
109		75		70
110		76		71
111				72
117				75
118				76
119				77
120				78
122				79
123				80

n/k	6	7	8	9
136				89
137				90
138	99			91
139				92
142			97	
143			98	
144			99	
145			100	
148		103		
149		104		
157			107	
158			108	
162		113	111	
163		114	112	
164		115		
165		116		
167		117		
168		118		
169		119		
170		120		
173		121	119	
174		122	120	
175		123		
176	127	124		
177	128			
178		125		
179		126	123	
180		127	124	
182			125	
183			126	
184			127	
185			128	
188			129	
189			130	
190			131	
191			132	
192	139			
193	140		133	
194			134	

[3]Fett gedruckte Einträge sind optimal.

n/k	6	7	8	9
195			135	
196			136	
199			137	
200			138	
201			139	136
202			140	
204			141	
205			142	
206			143	
207			144	
210			145	
211			146	
212			147	
213		151	148	
214		152		145
215			149	146
216			150	147
217			151	148
218		155	152	
219		156		
221			153	150
222			154	151
223			155	152
224			156	
225				153
226			157	154
227			158	155
228			159	156
229			160	
231				157
232				158
233				159
234				160
253			177	
254			178	
255			179	
256			180	

Tabelle C.3.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_4$.

C.4. Ergebnisse[4] über $\mathbb{F}_5$

n/k	5	6	7	8
16			**8**	
25			**15**	
26			**16**	
29				16
33			21	
34			22	
35			23	
36			**24**	
37				22
43				26
44				27
49		34		
52				33
53			35	
54			36	
55				35
56				36
57			38	
58			39	
59				38
60				39
71		51		
75		54		
76		55		51
79		57		
80		58		54
81		59		
82	62	60		
83	63			
84	64			
85	**65**			
90		66		
91		67		
92		68		
93		69		

[4]Fett gedruckte Einträge sind optimal.

n/k	5	6	7	8
94		70		
100		75		

Tabelle C.4.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_5$.

C.5. Ergebnisse[5] über $\mathbb{F}_7$

n/k	4	5	6
24		17	
27	**21**		
30		22	
33			23
36		27	
43			31
47		36	
48		37	
53		41	
54		42	
55			41
56			42
58			43
59			44
60		47	45
65			49
66		52	
68			51
69			52
70			53
71			54
72			55
79		63	
82			63
85		68	
88			68
91		73	
95		76	
96		77	
97		78	
98		79	
99		80	77
100		81	78

Tabelle C.5.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_7$.

[5] Fett gedruckte Einträge sind optimal.

C.6. Ergebnisse über $\mathbb{F}_8$

n/k	5	6
25		17
26	19	
30		21
41		30
52	41	
59		45
73	59	
74	60	
84	68	
85	69	
86	70	
87	71	
88	72	
89	73	
90	74	71
91		72
95	78	
96		76
97		77
99	81	
100	82	
101	83	
102	84	
122	101	
123	102	

Tabelle C.6.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_8$.

C.7. Ergebnisse über $\mathbb{F}_9$

n/k	4	5
41		32
56		45
63		51
71		58
78		64
92		76
93		77
94		78
100		83
101		84
106		88
107		89
108	93	90
109	94	91
110		92
114		95
115		96
116		97
117		98
129		108
130		109

Tabelle C.7.: Minimaldistanz der mit *Solver* gefundenen neuen Codes über $\mathbb{F}_9$.

Literaturverzeichnis

[ARC02] Aydin, Noah ; Ray-Chaudhuri, Dijen: Quasi-cyclic codes over Z4 and some new binary codes. In: *IEEE Transactions on Information Theory* 48 (2002), S. 2065–2069

[Asa] Asamov, Tsvetan: *The Database for Z4 Codes*. `http://asamov.com/Z4Codes`, Abruf: 17.01.2011

[BBF+06] Betten, Anton ; Braun, Michael ; Fripertinger, Harald ; Kerber, Adalbert ; Kohnert, Axel ; Wassermann, Alfred: *Error-Correcting Linear Codes*. Berlin : Springer Verlag, 2006 (Algorithms and Computation in Mathematics)

[BKW05] Braun, Michael ; Kohnert, Axel ; Wassermann, Alfred: Optimal Linear Codes From Matrix Groups. In: *IEEE Transactions on Information Theory* 51 (2005), S. 4247–4251

[Bra04] Braun, Michael: Construction of linear codes with large minimum distance. In: *IEEE Transactions on Information Theory* 50 (2004), S. 1687–1691

[CD73] Clark, William E. ; Drake, David A.: Finite Chain Rings. In: *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg* 39 (1973), S. 147–153

[CH97] Constantinescu, Ioana ; Heise, Werner: A metric for codes over residue class rings. In: *Problems of Information Transmission* 33 (1997), S. 208–213

[CK] Cameron, Peter J. ; Kantor, William M.: *Antiflag-transitive collineation groups revisited*. `http://www.maths.qmul.ac.uk/~pjc/odds/antiflag.pdf`, Abruf: 17.01.2011

[Com] Computational Algebra Group, The: *Magma Computational Algebra System*. `http://magma.maths.usyd.edu.au/magma`, Abruf: 17.01.2011

[Es98] Etzion, Tuvi ; Östergård, Patric R. J.: Greedy and Heuristic Algorithms for Codes and Colorings. In: *IEEE Transactions on Information Theory* 44 (1998), S. 382–388

[Feu09] Feulner, Thomas: The Automorphism Groups of Linear Codes and Canonical Representatives of Their Semilinear Isometry Classes. In: *Advances in Mathematics of Communications* 3 (2009), S. 363–383

[GAP] GAP Group, The: *GAP System for Computational Discrete Algebra*. `http://www.gap-system.org`, Abruf: 17.01.2011

[Gra] Grassl, Markus: *Bounds on the minimum distance of linear codes and quantum codes*. `http://www.codetables.de`, Abruf: 17.01.2011

[GS99] Greferath, Marcus ; Schmidt, Stefan E.: Gray Isometries for Finite Chain Rings and a Nonlinear Ternary $(36, 3^{12}, 15)$ Code. In: *IEEE Transactions on Information Theory* 47 (1999), S. 2522–2524

[GS00] Greferath, Marcus ; Schmidt, Stefan E.: Finite-Ring Combinatorics and MacWilliams' Equivalence Theorem. In: *Journal of Combinatorial Theory Series A* 92 (2000), S. 17–28

[HHL00] Hemme, Ludger ; Honold, Thomas ; Landjev, Ivan: Arcs in projective Hjelmslev spaces obtained from Teichmüller sets. In: *Seventh international workshop on algebraic and combinatorial coding theory (ACCT-7)*. Bansko, Bulgaria, 18.-24. Juni 2000, S. 177–182

[HK99] Hill, Ray ; Kolev, Emil: A survey of recent results on optimal linear codes. In: *Combinatorial Designs and their Applications*. Boca Raton : Chapman&Hall/CRC, 1999 (Research Notes in Mathematics, Bd. 403), S. 127–152

[HKC+94] Hammons, A. Roger Jr. ; Kumar, P. Vijay ; Calderbank, A. Robert ; Sloane, Neil J. A. ; Solé, Patrick: The Z4-linearity of Kerdock, Preparata, Goethals, and Related Codes. In: *IEEE Transactions on Information Theory* 40 (1994), S. 301–319

[HL00] Honold, Thomas ; Landjev, Ivan: Linear Codes over Finite Chain Rings. In: *Electronic Journal of Combinatorics* 7 (2000)

[HL05] Honold, Thomas ; Landjev, Ivan: On maximal arcs in projective Hjelmslev planes over chain rings of even characteristic. In: *Finite Fields and their Applications* 11 (2005), S. 292–304

[HP03] Huffman, W. Cary ; Pless, Vera: *Fundamentals of Error-Correcting Codes*. Cambridge : Cambridge University Press, 2003

[HW99] Hemme, Ludger ; Weijand, Daniel: *Arcs in projektiven Hjelmslev-Ebenen*, Technische Universität München, Fortgeschrittenenpraktikum, 1999

[Ker72] Kerdock, Anthony M.: A class of low-rate non-linear binary codes. In: *Information and Control* 20 (1972), S. 182–187

[Kie06] Kiermaier, Michael: *Arcs und Codes über endlichen Kettenringen*, Technische Universität München, Diplomarbeit, 2006

[KM76] Kramer, Earl S. ; Mesner, Dale M.: t-Designs on hypergraphs. In: *Discrete Mathematics* 15 (1976), S. 263–296

[KN94] Kuzmin, Alexey S. ; Nechaev, Alexander A.: Linearly representable codes and Kerdock code over an arbitrary Galois field of the characteristic 2. In: *Russian Mathematical Surveys* 49 (1994), S. 183–184

[KN96] Kuzmin, Alexey S. ; Nechaev, Alexander A.: Linearly presentable codes. In: *Proceedings of the International Symposium on Information Theory and its Applications (ISITA)* (1996), S. 31–34

[KN97] Kuzmin, Alexey S. ; Nechaev, Alexander A.: Trace-Function on a Galois Ring in Coding Theory. In: *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes: 12th International Symposium (AAECC-12), Toulouse*. Berlin/Heidelberg : Springer Verlag, 1997 (Lecture Notes in Computer Science, Bd. 1255), S. 277–290

[Knu00] Knuth, Donald E.: Dancing links. In: *Millennial Perspectives in Computer Science: Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare*, Palgrave Macmillan, 2000, S. 187–214

[Koh] Kohnert, Axel: *Best Linear Codes*. `http://www.algorithm.uni-bayreuth.de/en/research/Coding_Theory/Linear_Codes_BKW/index.html`, Abruf: 17.01.2011

[Kru24] Krull, Wolfgang: Algebraische Theorie der Ringe II. In: *Mathematische Annalen* 91 (1924), S. 1–46

[KZ] Kiermaier, Michael ; Zwanzger, Johannes: A Z4-linear code of high minimum Lee distance derived from a hyperoval. Akzeptiert für Veröffentlichung in *Advances in Mathematics of Communications*.

[KZ09] Kohnert, Axel ; Zwanzger, Johannes: New linear codes with prescribed group of automorphisms found by heuristic search. In: *Advances in Mathematics of Communications* 3 (2009), S. 157–166

[KZ10] Kiermaier, Michael ; Zwanzger, Johannes: A new series of Z4-linear codes of high minimum Lee distance derived from the Kerdock code. In: *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*. Budapest, 5.-9. Juli 2010, S. 929–932

[Lam01] Lam, Tsit Y.: *A First Course in Noncommutative Rings*. Berlin : Springer Verlag, 2001

[Lüb] Lübeck, Frank: *Conway polynomials for finite fields*. `http://www.math.rwth-aachen.de/~Frank.Luebeck/data/ConwayPol/index.html`, Abruf: 17.01.2011

[Lit98] Litsyn, Simon: An updated table of the best binary codes known. In: Pless, Vera (Hrsg.) ; Huffman, W. Cary (Hrsg.): *Handbook of Coding Theory* Bd. 1. Amsterdam : Elsevier, 1998, S. 463–498

[LRS] Litsyn, Simon ; Rains, Eric M. ; Sloane, Neil J. A.: *Table of Nonlinear Binary Codes*. `http://www.eng.tau.ac.il/~litsyn/tableand/index.html`, Abruf: 17.01.2011

[McD74] McDonald, Bernard R.: *Finite rings with identity*. New York : M. Dekker, 1974

[Nec73] Nechaev, Alexander A.: Finite principal ideal rings. In: *Mathematics of the USSR-Sbornik* 20 (1973), S. 364–382

[Nec91] Nechaev, Alexander A.: Kerdock code in a cyclic form. In: *Discrete Mathematics and Applications* 1 (1991), S. 365–384

[NR67] Nordstrom, Alan W. ; Robinson, John P.: An optimum nonlinear code. In: *Information and Control* 11 (1967), S. 613–616

[NS00] Norton, Graham H. ; Sălăgean, Ana: On the Structure of Linear and Cyclic Codes over a Finite Chain Ring. In: *Applicable Algebra in Engineering, Communication and Computing* 10 (2000), S. 489–506

[Par90] Parker, Richard: *Finite Fields and Conway Polynomials*. 1990. – Vortrag am IBM Scientific Center Heidelberg

[Pre68] Preparata, F. P.: A class of optimum nonlinear double-error-correcting codes. In: *Information and Control* 13 (1968), S. 378–400

[Sch92] Scheerhorn, Alfred: Trace- and Norm-Compatible Extensions of Finite Fields. In: *Applicable Algebra in Engineering, Communication and Computing* 3 (1992), S. 199–209

[Woo99] Wood, Jay A.: Duality for modules over finite rings and applications to coding theory. In: *American Journal of Mathematics* 121 (1999), S. 555–575

[WZ10] Wassermann, Alfred ; Zwanzger, Johannes: *Strenghtening the upper bounds on linear codes with lattice point enumeration*. September 2010. – Vortrag auf dem IEEE Information Theory Workshop (ITW) in Dublin

[ZL87] Zinoviev, Victor A. ; Litsyn, Simon: On the general construction of codes' shortening. In: *Problems of Information Transmission* 23 (1987), S. 111–116

[Zwa] Zwanzger, Johannes: *Linear codes over finite chain rings*. `http://www.mathe2.uni-bayreuth.de/20er/codedb`, Abruf: 17.01.2011

[Zwa07] Zwanzger, Johannes: *Computergestützte Konstruktion von linearen Codes mit hoher Minimaldistanz unter Verwendung heuristischer Methoden*, Universität Bayreuth, Diplomarbeit, 2007

[Zwa08] Zwanzger, Johannes: A Heuristic Algorithm for the Construction of Good Linear Codes. In: *IEEE Transactions on Information Theory* 54 (2008), S. 2388–2392

Erklärung

Hiermit erkläre ich, dass ich diese Doktorarbeit selbst verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Sie wurde noch in keinem anderen Prüfungsverfahren vorgelegt.

Neubiberg, den 24. Juni 2011

(Johannes Zwanzger)