UNIVERSITÄT
BAYREUTH

Fakultät für Mathematik, Physik und Informatik

Computergestützte Berechnung eines eindeutigen Repräsentanten der semilinearen Isometrieklasse eines fehlerkorrigierenden, linearen Codes und Bestimmung der Automorphismengruppe

Diplomarbeit im Fach Mathematik

von

Thomas Feulner

Wolfsgraben 1

95488 Eckersdorf

Abgabetermin: 4. Januar 2008

Themensteller: Professor Dr. Kerber

Inhaltsverzeichnis

Einleitung

Aufgabe der Codierungstheorie ist es Verfahren bereitzustellen, um Fehler bei der Datenübertragung ohne Rückfragemöglichkeit über ein fehlerbehaftetes Übertragungssystem zu erkennen und vor allem zu korrigieren. Beispiele solcher Übertragungswege sind das Lesen einer CD oder die Übermittlung von Bildern eines Satelliten zur Erde. Hierzu werden die Daten durch Hinzufügen redundanter Informationen bereits vor der Übertragung so verändert, dass Übertragungsfehler bis zu einem gewissen Grad korrigiert werden können. Ziel ist es möglichst wenig redundante Information zu generieren und trotzdem eine Korrektur möglichst vieler Fehler zu ermöglichen.

Eine wichtige Klasse von Codes bilden dabei die linearen Codes über endlichen Körpern. Sie stellen zum einen effiziente Codier– und Decodierverfahren zur Verfügung, sind leicht beschreibbar und lassen eine Vielzahl an Aussagen aufgrund ihrer algebraischen Struktur zu.

Ein linearer (n, k, q) – Code C ist ein k-dimensionaler Unterraum des $GF(q)^n$. Dieser Unterraum lässt sich durch Auswahl von k linear unabhängigen Vektoren auch als Bild einer linearen Abbildung beschreiben. Eine diese Abbildung repräsentierende Matrix Γ nennt man Generatormatrix des Codes. Natürlich gibt es viele Möglichkeiten k linear unabhängige Vektoren in C auszuwählen und als Matrix anzuordnen. Will man nun den Code durch seine Generatormatrix in einer Datenbank speichern, so muss man sich an dieser Stelle bereits für eine eindeutige Form entscheiden.

Neben der Auswahl der Basis stellt sich noch ein weiteres Problem. Es gibt zumeist eine Vielzahl linearer Codes, die sich in ihrer fehlerkorrigierenden Eigenschaft nicht unterscheiden. Für die Anwendung genügt es, einen Repräsentanten einer jeden Klasse vorliegen zu haben. Ein Fehler in der Übertragung sei hierbei eine veränderte Koordinatenstelle des gesandten Vektors $c \in C$. Zwei Codes sind in dieser Hinsicht als äquivalent anzusehen, wenn es eine Abbildung gibt, welche die Hammingdistanz erhält und die Codes aufeinander abbildet. Solche Abbildungen des $GF(q)^n$ auf sich selbst nennt man Isometrien, die Codes isometrisch. Um nun zum Beispiel Forschungsergebnisse vergleichen und vollständige Kataloge verwalten zu können, ist es notwendig, eindeutige Repräsentanten zu generieren und vorliegende Codes auf Isometrie zu testen.

Da beliebige Isometrien, einen linearen Code auch auf nichtlineare Teilmengen des $GF(q)^n$ – sogenannte Blockcodes – abbilden können, beschränken wir uns auf Isometrien, welche Unterräume auf Unterräume abbilden, d.h. lineare Codes auf lineare Codes mit den gleichen Parametern n und k. Für $n \geq 3$ sind dies gerade die Isometrien, die

gleichzeitig semilineare Abbildungen sind[1]. Eine Abbildung $\sigma : GF(q)^n \to GF(q)^n$ ist semilinear, falls es einen Körperautomorphismus α gibt, so dass für alle $u, v \in GF(q)^n$ und $\lambda \in GF(q)$ gilt:

$$\sigma(u+v) = \sigma(u) + \sigma(v) \quad \text{und} \quad \sigma(\lambda v) = \alpha(\lambda)\sigma(v).$$

Unter dieser Einschränkung können wir die Äquivalenzklassen semilinear isometrischer (n, k, q) – Codes auch als Bahnenmenge

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes \big(Gal(q) \times S_n\big)\Big) \backslash\backslash GF(q)^{k \times n}$$

einer Gruppenoperation ausdrücken und Aussagen der Gruppentheorie zum Einsatz bringen. Aus diesem Grunde führt das erste Kapitel bereits tiefer in die Gruppentheorie ein. Weiter dient es zur Festlegung der Notation und einer Einleitung in das benötigte Wissen der Codierungstheorie.

In Kapitel 2 wird ein allgemeiner Algorithmus zur Berechnung eines eindeutigen Repräsentanten der Bahn $G(x), x \in X$ einer Gruppenoperation ${}_GX$ schrittweise entwickelt. Das dort entwickelte Verfahren kann als Verallgemeinerung des Kanonisierungsalgorithmus für Graphen [McK81] gesehen werden und greift zur Effizienzsteigerung auf G-Homomorphismen zurück. Weiterhin liefert es zusätzlich als Ausgabe die Automorphismengruppe $Aut(x)$.

Eine direkte Anwendung dieses Algorithmus auf obige Operation der semilinearen Isometrien auf den Generatormatrizen linearer Codes ist jedoch wegen der Gruppenordnung nicht sinnvoll. Das Kapitel 3 führt daher das Problem auf die Operation der symmetrischen Gruppe S_n auf der Menge

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k \times n}$$

über. In Kapitel 4 rechtfertigen wir diesen Schritt, indem wir zeigen, dass die Operation von $\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes Gal(q)\Big)$ auf $GF(q)^{k \times n}$ ohne aufwendigen Kanonisierungsalgorithmus, sondern durch „Minimierung“ der Spalten der vorliegenden Generatormatrix realisiert werden kann. Desweiteren beschreiben wir in diesem Kapitel, welche Homomorphismen zum Einsatz gebracht werden.

Das Kapitel 5 beschreibt die im Zuge der Diplomarbeit entstandene Software. Das Ende der Arbeit bildet einen Vergleich mit weiteren Algorithmen aus diesem Bereich. Weiter wird ein kurzer Ausblick zur ringtheoretischen Codierungstheorie gegeben. Neuere Ansätze in der Codierungstheorie betrachten Untermoduln von ${}_RR^n$ eines endlichen Kettenrings R. Auch diese lassen sich mit Hilfe von Generatormatrizen bzw. linearen Abbildungen darstellen. Der in dieser Diplomarbeit entwickelte Algorithmus sollte sich auch auf diese Situation durch entsprechende ringtheoretische Anpassungen des Kapitels 4 verallgemeinern lassen.

[1] Codes $C \leq GF(q)^n$ mit $n < 3$ sind nicht in der Lage Fehler zu korrigieren und sind daher ohnehin für die Codierungstheorie nicht von Interesse

1. Notationen und Grundlagen

In diesem ersten Kapitel sollen nun zunächst die grundlegenden Definitionen und Begriffe der Codierungstheorie eingeführt werden. Um jedoch geeignete Formulierungen für die Isometriegruppen zur Verfügung zur Stellen, werden wir zunächst einen Überblick über das benötigte Wissen aus der Gruppentheorie geben. Ein Grundwissen des Lesers aus dem Bereich der Codierungstheorie und Algebra, insbesondere der Gruppentheorie, wird vorausgesetzt.

1.1. Gruppentheorie

Die in dieser Arbeit auftretenden Gruppen seien stets multiplikativ geschrieben und endlich.

1.1.1. Operationen von Gruppen auf Mengen

1.1.1 Definition (Operationen von Gruppen auf Mengen)**.** Sei G eine Gruppe und X eine nicht leere Menge. Man sagt, G operiert auf X oder X ist eine G–Menge, falls eine Abbildung $G \times X \to X, (g, x) \mapsto gx$ existiert, mit

$$g'(gx) = (g'g)x \text{ und } 1_G x = x \quad \forall g, g' \in G, x \in X.$$

Wir schreiben auch kurz ${}_GX$, falls es so eine Abbildung gibt, um anzudeuten, dass G von links operiert. Analog kann man auch eine Operation von rechts definieren.
Durch die Definition

$$x \sim_G x' :\Longleftrightarrow \exists g \in G : gx = x'$$

induziert jede Gruppenoperation eine Äquivalenzrelation auf X. Wir nennen x und x' auch G–abhängig. Mit $G(x) := \{gx \mid g \in G\}$ bezeichnen wir die Bahn des Elements x bzw. die Äquivalenzklasse von x der induzierten Relation. Die Menge aller Bahnen von G auf X sei $G\backslash\backslash X := \{G(x) \mid x \in X\}$. Die Menge aller Bahnen bildet somit eine Mengenpartition von X. Ein minimales, aber vollständiges Repräsentantensystem T von $G\backslash\backslash X$ heißt Transversale von $G\backslash\backslash X$. Es gelte also:

$$X = \dot{\bigcup}_{t \in T} G(t).$$

Zu $g \in G$ bezeichne $X_g := \{x \in X \mid gx = x\}$ die Menge der Fixpunkte von g.

1.1.2 Beispiel. 1. Die Multiplikation $v \cdot A$ eines Vektors $v \in GF(q)^n$ mit einer invertierbaren Matrix A von rechts, ist Beispiel für die Operation einer Gruppe $G = GL_n(q)$ auf einer Menge $X = GF(q)^n$ von rechts.

2. Ist X eine endliche Menge, auf der die endliche Gruppe G von links operiert, Y eine nicht leere, endliche Menge, dann induziert die gegebene Operation ${}_GX$ eine Operation ${}_G(Y^X)$ von G auf der Menge aller Abbildungen von X nach Y mit Hilfe der Definition:

$$G \times Y^X \to Y^X : (g, f) \mapsto f \circ \overline{g}^{-1},$$

wobei $\overline{g} := \delta(g)$ das Bild des Gruppenhomomorphismus $\delta : G \to S_X, g \mapsto \overline{g}$ mit $\overline{g} : x \mapsto gx$ sei.
Ist die Funktion $\widetilde{f} = gf$ so, ist also $\widetilde{f}(x) = f(g^{-1}(x)), \forall x \in X$.

1.1.3 Definition. Eine Gruppenoperation ${}_GX$ heißt endlich, falls sowohl G als auch X endlich sind. Ist X eine endliche Menge, so ist die Gruppenoperation ${}_{\overline{G}}X$ endlich. Dabei sei $\overline{G} = \delta(G) = \{\overline{g} \mid g \in G\}$ das Bild des oben eingeführten Gruppenhomomorphismus $\delta : G \to S_X, g \mapsto \overline{g}$. Als Untergruppe der endlichen Gruppe S_X ist $\overline{G}$ trivialerweise endlich.

1.1.4 Definition (Stabilisatoren)**.** Sei eine beliebige Gruppenoperation ${}_GX$ gegeben. Zu jedem $x \in X$ heißt die Untergruppe $G_x := \{g \in G \mid gx = x\}$ der Stabilisator von x. Als Verallgemeinerung hierzu definieren wir für eine Teilmenge $Y \subseteq X$ den mengenweisen Stabilisator $G_Y := \{g \in G \mid gY = Y\}$. Es gilt $G_{\{x\}} = G_x$ und G_Y ist eine Untergruppe von G.
Im Gegensatz hierzu definiert man den punktweisen Stabilisator von k verschiedenen Elementen $x_0, \ldots, x_{k-1} \in X$ als den Stabilisator der Folge $(x_0, \ldots, x_{k-1}) \in X^k_{inj}$. Dabei ist die Reihenfolge der Elemente beliebig, denn es gilt:

$$G_{(x_0,\ldots,x_{k-1})} = \bigcup_{i=0}^{k-1} G_{x_i}.$$

1.1.5 Hilfssatz (Eindeutigkeit der Darstellung)**.** *Ist T eine Transversale von $G\backslash\backslash X$, so ist jedes $x \in X$ bis auf Linksmultiplikation mit Elementen des Stabilisators G_x eindeutig als $x = g_x t_x$ darstellbar.*

Beweis. T ist Transversale, insbesondere gilt $X = \dot{\bigcup}_{t \in T} G(t)$. Somit existiert eine solche Darstellung zu jedem $x \in X$. Ist nun $x = g_x t_x = g'_x t'_x$, so ist $t_x = g_x^{-1} g'_x t'_x$, also liegen t_x und t'_x in der gleichen Bahn. Da aber T minimal ist, muss $t_x = t'_x$ gelten.
Es folgt $g_x^{-1} x = g_x'^{-1} x$ und damit $x = g'_x g_x^{-1} x$. Die beiden Darstellung unterscheiden sich daher nur durch Linksmultiplikation mit dem Stabilisatorelement $g'_x g_x^{-1}$. □

1.1.6 Definition (Kanonisierende Abbildung)**.** Sei ${}_GX$ eine Gruppenoperation und T eine Transversale von $G\backslash\backslash X$. Eine Abbildung $\gamma : X \to G, x \mapsto g_x^{-1}$ mit $x = g_x t_x$ heißt kanonisierende Abbildung zu T. Das Element t_x heißt kanonischer Repräsentant der Bahn $G(x)$.

1.1.7 Definition (Basis und Stabilisatorkette)**.** Sei ${}_GX$ eine endliche Gruppenoperation. Ein Tupel $(x_0, \ldots, x_{k-1}) \in X^k_{inj}, k \in n = |X|$ mit $G_{(x_0,\ldots,x_{k-1})} = \{\mathrm{id}\}$ heißt Basis. ${}_GX$ heißt treu, falls es in X Basen gibt.
Ist $(x_0, \ldots, x_{k-1})$ eine Basis und $G^{(i)} := G_{(x_0,\ldots,x_{i-1})}$ der punktweise Stabilisator der ersten i Elemente der Basis, so heißt die Untergruppenkette $G = G^{(0)} \geq G^{(1)} \geq \ldots \geq G^{(k-1)} \geq G^{(k)} = \{\mathrm{id}\}$ Stabilisatorkette von G zur Basis $(x_0, \ldots, x_{k-1})$.

1.1.8 Satz. *Sei ${}_GX$ eine endliche, treue Gruppenoperation und $(x_0, \ldots, x_{k-1})$ eine Basis sowie $G^{(i)}$ wie oben definiert. Desweiteren sei ein Satz von Transversalen $T^{(i)}$ der Linksnebenklassen $G^{(i)}/G^{(i+1)}, i \in k$ gegeben. Es gilt:*

$$G^{(i)} = T^{(i)} \cdot \ldots \cdot T^{(k-1)}$$

1.1.9 Folgerung. *Zwei Elemente $t, t' \in G^{(i)}$ liegen genau dann in der gleichen $G^{(i+1)}$–Nebenklasse, falls $tx_i = t'x_i$.*
Für die Transversale $T^{(i)}$ sind also die Elemente $tx_i, t \in T^{(i)}$ verschieden. Insbesondere ist also $\left|T^{(i)}\right| < k - i$, da $\{tx_i \mid t \in T^{(i)}\} \subseteq \{x_i, \ldots, x_{k-1}\}$. Ist $g \in G^{(i)}$ beliebig, so gibt es genau ein $t \in T^{(i)}$ mit $gx_i = tx_i$.

1.1.10 Definition. Sei G eine Gruppe, die auf den Mengen X und Y operiere. Eine Abbildung $\varphi : X \to Y$ heißt G–Homomorphismus, falls sie verträglich mit der Gruppenoperation ist. Das heißt

$$\forall x \in X\ \forall g \in G : \varphi(gx) = g\varphi(x).$$

1.1.11 Hilfssatz. *Seien zwei Gruppenoperationen ${}_GX$ und ${}_HY$, sowie ein Homomorphismus $\varphi : G \to H$ und eine Abbildung $\theta : X \to Y$ mit $\theta(gx) = \varphi(g)\theta(x)$ für alle $g \in G$ und $x \in X$ gegeben. Dann gilt für jedes $x \in X$:*

$$G_x \leq \varphi^{-1}(H_{\theta(x)}) \leq G.$$

Beweis. Sei $x \in X$ und $g \in G_x$ beliebig. Es gilt:

$$\theta(x) = \theta(gx) = \varphi(g)\theta(x),$$

also ist $g \in \varphi^{-1}(H_{\theta(x)})$.
Bleibt zu zeigen, dass $\varphi^{-1}(H_{\theta(x)}) \leq G$. Seien dazu $g, g' \in \varphi^{-1}(H_{\theta(x)})$.

$$\varphi(gg')\theta(x) = \varphi(g)\varphi(g')\theta(x) = \varphi(g)\theta(x) = \theta(x).$$

Damit ist aber $\varphi(gg') \in H_{\theta(x)} \Rightarrow gg' \in \varphi^{-1}(H_{\theta(x)})$.
Wegen der vorausgesetzten Endlichkeit der auftretenden Gruppen genügt die Abgeschlossenheit bereits um die Gruppeneigenschaft zu prüfen. □

1.1.12 Satz (Homomorphieprinzip für Gruppenoperationen)**.** *Gegeben seien zwei Gruppenoperationen ${}_GX$ und ${}_HY$, sowie ein Gruppenepimorphismus*[1] $\varphi : G \to H$ *und eine surjektive Abbildung* $\theta : X \to Y$ *mit* $\theta(gx) = \varphi(g)\theta(x)$ *für alle* $g \in G$ *und* $x \in X$*. Weiterhin sei eine Transversale* $T_{H\backslash\backslash Y}$ *bereits bestimmt. Es gilt:*

1. *Jede Bahn* $\omega \in G\backslash\backslash X$ *schneidet genau eines der Urbilder* $\theta^{-1}(y)$ *der Elemente* $y \in T_{H\backslash\backslash Y}$*.*
2. $\theta^{-1}(y)$ *zerfällt in Bahnen von* $\varphi^{-1}(H_y)$ *und zwei verschiedene* $\varphi^{-1}(H_y)$ *– Bahnen auf* $\theta^{-1}(y)$ *liegen in verschiedenen Bahnen von G auf X.*
3. *Seien* $T_{\varphi^{-1}(H_y)\backslash\backslash\theta^{-1}(y)}, y \in T_{H\backslash\backslash Y}$ *Transversalen der Urbilder unter der Operation des Urbilds des zugehörigen Stabilisators. Dann erhält man eine Transversale* $T_{G\backslash\backslash X}$ *von* $G\backslash\backslash X$ *als disjunkte Vereinigung obiger Transversalenmengen.*

$$T_{G\backslash\backslash X} := \dot{\bigcup}_{y \in T_{H\backslash\backslash Y}} T_{\varphi^{-1}(H_y)\backslash\backslash\theta^{-1}(y)}.$$

Beweis. Zunächst zu 1.: Sei $\omega \in G\backslash\backslash X$ beliebig.

- Sei $x \in \omega$ beliebig. Die Bahn ω hat somit einen nicht leeren Schnitt mit $\theta^{-1}(y')$, für $y' = \theta(x)$.
- Sei y der Repräsentant der Bahn $H(y')$ und $y = hy'$ für ein geeignetes $h \in H$. Da φ surjektiv ist, existiert ein $g \in G$ mit $\varphi(g) = h$. Für das weitere Bahnenelement $gx \in \omega$ folgt:

  $$\theta(gx) = \varphi(g)\theta(x) = hy' = y,$$

  also:

  $$\omega \cap \theta^{-1}(y) \neq \emptyset.$$

- Weiterhin ist das Bild $\theta(\bar{g}x)$ jedes $\bar{g}x \in \omega$ ein Element der Bahn von y, somit kann nur der Schnitt mit dem Urbild von y nicht leer sein.

Zu 2.: Für ein beliebiges $x \in \theta^{-1}(y)$ und $g \in \varphi^{-1}(H_y)$ ist auch $gx \in \theta^{-1}(y)$:

$$\theta(gx) = \varphi(g)\theta(x) = \varphi(g)y = y.$$

Seien $x, x' \in \theta^{-1}(y)$ mit $x' = gx$ und $g \in G$. Dann gilt:

$$y = \theta(x') = \theta(gx) = \varphi(g)\theta(x) = \varphi(g)y.$$

Dies ergibt $g \in \varphi^{-1}(H_y)$. Also sind G–abhängige Elemente in $\theta^{-1}(y)$ auch $\varphi^{-1}(H_y)$ – abhängig. Die Kontraposition ergibt die Behauptung.
Zu 3.: Zusammenfassung der Ergebnisse aus 1. und 2. □

[1] surjektiver Gruppenhomomorphismus

1.1.13 Folgerung. *Seien die Voraussetzungen wie im Homomorphieprinzip. Dann sind zwei Elemente $x, x' \in X$ genau dann G–abhängig, falls die folgenden zwei Bedingungen erfüllt sind:*

- *$\theta(x)$ und $\theta(x')$ sind H-abhängig,*
- *für ein solches beliebiges $h \in H$ mit $\theta(x) = h\theta(x')$ und $g \in \varphi^{-1}(h)$ gilt: x und gx' sind $\varphi^{-1}(H_{\theta(x)})$ – abhängig.*

1.1.14 Folgerung. *Sei θ zusätzlich zu obigen Voraussetzungen außerdem bijektiv. Wir erhalten mit*

$$\Psi : G\backslash\backslash X \to H\backslash\backslash Y, G(x) \mapsto H(\theta(x))$$

eine wohldefinierte Bijektion zwischen den G–Bahnen auf X und der Menge der H–Bahnen auf Y. Für die Bahn $H(\theta(x))$ eines beliebigen $x \in X$ gilt:

$$H(\theta(x)) = \theta(G(x)).$$

Beweis. Wir zeigen, dass der zweite Punkt in obiger Folgerung unter diesen Voraussetzungen überflüssig ist. Denn sind $x, x' \in X$ und $h \in H$ mit $\theta(x) = h\theta(x')$ gegeben, so folgt für jedes beliebiges $g \in \varphi^{-1}(h)$:

$$\theta(gx') = \varphi(g)\theta(x') = h\theta(x') = \theta(x).$$

Und somit ist $gx' = x$ wegen der Bijektivität von θ. Die Aussage vereinfacht sich damit zu:

$$G(x) = G(x') \iff H(\theta(x))) = H(\theta(x')).$$

Damit ist Ψ wohldefiniert und injektiv, die Surjektivität folgt sofort aus der Surjektivität von θ.
Bleibt die Gleichung für die Bahn:

$$\theta(G(x)) = \theta\big(\{gx \mid g \in G\}\big) = \big\{\theta(gx) \mid g \in G\big\} = \big\{\varphi(g)\theta(x) \mid g \in G\big\} \overset{\varphi \text{ surjektiv}}{=} \big\{h\theta(x) \mid h \in H\big\} = H\big(\theta(x)\big).$$

□

1.1.15 Bemerkung. Gegeben seien zwei Gruppenoperationen ${}_GX$ und ${}_HY$, sowie ein Gruppenhomomorphismus $\varphi : G \to H$. Dies induziert damit auch eine Operation von G auf Y via

$$G \times Y \to Y, \quad (g, y) \mapsto \varphi(g)y.$$

Beweis. Seien $g_1, g_2 \in G$ beliebig:

$$g_1(g_2 y) = g_1(\varphi(g_2)y) = \varphi(g_1)(\varphi(g_2)y) = \varphi(g_1 g_2)y = (g_1 g_2)y.$$

□

1.1.16 Definition (Kranzprodukt)**.** Sei ${}_G X$ Gruppenoperation und H eine weitere Gruppe. Die Gruppe $H \wr_X G$ hat als Grundmenge $H^X \times G = \{(\varphi; g) \mid \varphi : X \to H, g \in G\}$ und als Verknüpfung

$$(\varphi; g)(\varphi'; g') := (\varphi\varphi'_g; gg') \text{ mit } (\varphi\varphi'_g)(x) := \varphi(x) \cdot \varphi'(g^{-1}x), \forall x \in X.$$

1.1.17 Hilfssatz. *Operiert H auf einer Menge Y, G auf X, dann ist eine natürliche Gruppenoperation von $H \wr_X G$ auf Y^X gegeben durch:*

$$H \wr_X G \times Y^X \to Y^X : \big((\varphi; g), f\big) \mapsto \widetilde{f}, \text{ wobei } \widetilde{f}(x) := \varphi(x) f(g^{-1}x).$$

1.1.18 Hilfssatz. *1.* $1_{H \wr_X G} = (\epsilon; 1_G)$ *mit* $\epsilon : x \mapsto 1_H$,

2. $(\varphi; g)^{-1} = (\varphi^{-1}_{g^{-1}}; g^{-1})$ *wobei* $\varphi^{-1}_{g^{-1}}(x) := \varphi(gx)^{-1}$

Beweis. 1. trivial.

2. Wir multiplizieren mit dem angegebenen Gruppenelement:

$$(\varphi; g)(\varphi^{-1}_{g^{-1}}; g^{-1}) = (\varphi(\varphi^{-1}_{g^{-1}})_g; gg^{-1}) = (\epsilon; 1_G) = 1_{H \wr_X G},$$

denn für alle $x \in X$ gilt:

$$(\varphi(\varphi^{-1}_{g^{-1}})_g)(x) = \varphi(x) \cdot \varphi^{-1}_{g^{-1}}(g^{-1}x) = \varphi(x) \cdot \varphi\big(g(g^{-1}x)\big)^{-1} = \varphi(x) \cdot \varphi(x)^{-1} = 1_H.$$

□

1.1.2. Semidirekte Produkte von Gruppen

1.1.19 Definition (Das semidirekte Produkt)**.** Seien N und H Gruppen und $\theta : H \to Aut(N)$ ein Gruppenhomomorphismus. Das kartesische Produkt $N \times H$ wird durch Definition der Verknüpfung

$$\begin{aligned} \cdot : (N \times H) \times (N \times H) &\to N \times H \\ \big((n_0, h_0), (n_1, h_1)\big) &\mapsto \big(n_0 \theta(h_0)(n_1), h_0 h_1\big) \end{aligned}$$

zu einer Gruppe. Wir schreiben auch $N \rtimes_\theta H$ für die so gewonnen Gruppe und nennen es das (äußere) semidirekte Produkt von N und H.

1.1.20 Hilfssatz (Eigenschaften des semidirekten Produkts)**.** *Für das semidirekte Produkt $G := N \rtimes_\theta H$ gilt:*

1. $N \simeq N' := N \times \{1_H\} \trianglelefteq N \rtimes_\theta H$.
2. $G/N' \simeq H$, *der Isomorphismus ist definiert durch* $(h, n)N' \mapsto h$.
3. $H \simeq H' := \{1_N\} \times H \leq N \rtimes_\theta H$.
4. $N'H' = G$.

Beweis. Die Projektion auf die zweite Komponente $\Pi : N \rtimes_\theta H \to H, (n, h) \mapsto h$ ist ein Epimorphismus. Als Kern dieser Abbildung ist N' ein Normalteiler von G. Nach dem Homomorphiesatz für Gruppen ist weiterhin $G/Kern(\Pi) \simeq Bild(\Pi)$ also $G/N' \simeq H$. Der Isomorphismus ist aus dem Homomorphiesatz eindeutig vorgegeben durch $\psi : G/N' \to H, (n, h)N' \mapsto \Pi\big((n, h)\big) = h$.
Die Isomorphie von N und N' folgt sofort aus den Homomorphieeigenschaften von θ, da $\theta(1_H) = id_N$. Der Rest ist klar. □

1.1.21 Hilfssatz. *Sei G eine Gruppe, die auf einer Menge X operiere. N ein Normalteiler von G. ${}_NX$ die Einschränkung der Operation auf N. Dann gilt:*

1. *Für jede Bahn $N(x) \in N\backslash\backslash X$ und jedes beliebige $g \in G$ ist die Menge $gN(x)$ wieder ein Element von $N\backslash\backslash X$. Es gilt sogar: $gN(x) = N(gx)$.*
2. *Die Gruppe G operiert auf der Menge der N–Bahnen von X via*

$$G \times N\backslash\backslash X \to N\backslash\backslash X : (g, N(x)) \mapsto N(gx).$$

3. *Die Faktorgruppe G/N operiert auf $N\backslash\backslash X$ durch*

$$G/N \times N\backslash\backslash X \to N\backslash\backslash X : (gN, N(x)) \mapsto N(gx).$$

4. *Die Abbildungen*

$$\begin{aligned} &\Phi : G\backslash\backslash X \to G\backslash\backslash(N\backslash\backslash X), && G(x) \mapsto G(N(x)) \\ &\Psi : G\backslash\backslash X \to (G/N)\backslash\backslash(N\backslash\backslash X), && G(x) \mapsto (G/N)(N(x)) \end{aligned}$$

sind wohldefinierte Bijektionen der Bahnenmengen und für die Bahn eines $x \in X$ gilt:

$$\begin{aligned} G(x) &= \bigcup_{g \in G} g \cdot N(x) = \bigcup_{g \in G} N(gx), \\ G(x) &= \bigcup_{gN \in G/N} gN \cdot N(x) = \bigcup_{gN \in G/N} N(gx) \end{aligned}$$

Beweis. 1. $gN(x) = \{gnx \mid n \in N\} \overset{N \trianglelefteq G}{=} \{\overline{n}gx \mid \overline{n} \in N\} = N(gx) \in N\backslash\backslash X$

2. Die Abbildung ist wohldefiniert, denn für ein weiteres $x' \in N(x)$ gilt nach 1.:

$$N(gx') = gN(x') = gN(x) = N(gx).$$

Aus der Gleichung folgt weiter $1_G N(x) = N(x)$ und für beliebige $g_0, g_1 \in G$:

$$g_0(g_1 N(x)) = g_0 N(g_1 x) = N(g_0 g_1 x) = N((g_0 g_1)x) = (g_0 g_1)N(x)$$

3. Sei $g \in G$ beliebig, $g' = gn$ für ein $n \in N$

$$g'N \cdot N(x) = N(g'x) = g'N(x) = gnN(x) = gN(x) = N(gx) = gN \cdot N(x).$$

Die Zuordnung ist also unabhängig von der Wahl des Nebenklassenanführers. Analog zu oben sieht man auch, dass sie unabhängig von dem gewähltem Bahnenelement x ist.
Die Abbildung definiert eine Gruppenoperation, da

$$1_G N \cdot N(x) = N(x)$$

und für beliebige $gN, hN \in G/N$ folgendes gilt:

$$\begin{aligned} &gN \cdot (hN \cdot N(x)) = gN \cdot N(hx) = N(ghx) = \\ =&N((gh)x) = ((gh)N) \cdot N(x) = (gN \cdot hN) \cdot N(x). \end{aligned}$$

4. Die Abbildung Φ ist wohldefiniert und injektiv:

$$\begin{aligned} G(x) = G(x') &\iff \exists g \in G : gx = x' \\ &\iff \exists n \in N, g \in G : ngx = x' \\ &\iff \exists g \in G : N(gx) = N(x') \\ &\iff \exists g \in G : gN(x) = N(x') \\ &\iff G(N(x)) = G(N(x')) \end{aligned}$$

Für die Abbildung Ψ können wir analog beweisen:

$$\begin{aligned} G(x) = G(x') &\iff \exists g \in G : gx = x' \\ &\iff \exists n \in N, g \in G : ngx = x' \\ &\iff \exists g \in G : N(gx) = N(x') \\ &\iff \exists g \in G : gN \cdot N(x) = N(x') \\ &\iff (G/N)(N(x)) = (G/N)(N(x')) \end{aligned}$$

Die Surjektivität der beiden Abbildungen ist trivial.
Um die Gleichungen für die Bahnen zu beweisen, genügt es zu bemerken, dass

$$\bigcup_{gN \in G/N} gN = G = \bigcup_{g \in G} gN$$

und $N(gx) = gN(x) = \{gnx \mid n \in N\}$ gilt.

□

1.1.22 Folgerung. *Sei $G := N \rtimes_\theta H$ und ${}_GX$ eine Gruppenoperation. Dann operieren auch die Gruppen N und H auf X durch:*

$$N \times X \to X, (n, x) \mapsto (n, 1_H)x$$
$$H \times X \to X, (h, x) \mapsto (1_N, h)x.$$

Durch die Definition

$$H \times N\backslash\backslash X \to N\backslash\backslash X, (h, N(x)) \mapsto (1_N, h)N \cdot N(x) = N((1_N, h)x) = N(hx)$$

operiert H auf den N–Bahnen von X und es gibt eine Bijektion

$$\Phi : G\backslash\backslash X \to H\backslash\backslash(N\backslash\backslash X), G(x) \mapsto H(N(x))$$

der Menge aller G–Bahnen von X in die Menge der H–Bahnen von N–Bahnen auf X. Für die Bahn eines $x \in X$ gilt:

$$G(x) = \bigcup_{h \in H} N(hx).$$

Beweis. Seien N' und H' die oben definierten Untergruppen in G. Die Operationen von N bzw. H auf X sind entsprechend den Operationsvorschriften der isomorphen Gruppen N' bzw. H' definiert, und somit Gruppenoperationen. Da N' ein Normalteiler in G ist, können wir Hilfssatz 1.1.21 anwenden. Es ergibt sich

$$\Psi : G\backslash\backslash X \to (G/N')\backslash\backslash(N'\backslash\backslash X), G(x) \mapsto (G/N')(N'(x))$$

ist eine wohldefinierte Bijektion zwischen den Bahnenmengen. Nach Folgerung 1.1.14 ist $\theta : N'\backslash\backslash X \to N\backslash\backslash X, N'(x) \mapsto N(x)$ eine Bijektion, die nach Definition der Operation von H auf $N\backslash\backslash X$ verträglich mit dem Gruppenisomorphismus $\varphi : G/N' \to H, (n, h)N' \mapsto h$ ist. Wir wenden ein weiteres Mal Folgerung 1.1.14 an und erhalten die Bijektion:

$$\Omega : (G/N')\backslash\backslash(N'\backslash\backslash X) \to H\backslash\backslash(N\backslash\backslash X), (G/N')(N'(x)) \mapsto H(N(x))$$

Die Abbildung $\Phi = \Omega \circ \Psi$ ergibt sich durch Hintereinanderausführung von Ψ und Ω. Für die Bahn eines $x \in X$ selbst gilt:

$$G(x) = \bigcup_{(n,h)N' \in G/N'} (n, h)N' \cdot N'(x) = \bigcup_{(1_N,h)N' \in G/N'} (1_N, h)N' \cdot N'(x) =$$
$$\bigcup_{h \in H} N'((1_N, h)x) = \bigcup_{h \in H} N'(hx) = \bigcup_{h \in H} N(hx).$$

□

1.2. Codierungstheorie

Wir legen unseren Untersuchungen das folgende mathematische Modell zugrunde. Es sei ein endliches Alphabet Σ gegeben. Die Menge der übermittelbaren Worte sei Σ^k, wir nennen diese Menge auch den Nachrichtenraum. Eine injektive Abbildung $\gamma : \Sigma^k \to \Sigma^n$ führt eine Nachricht $v \in \Sigma^k$ in das Codewort $c = \gamma(v)$ über. Den Bildbereich $C := \gamma(\Sigma^k)$ nennt man den Code. Wir übertragen das Codewort über den Nachrichtenkanal und empfangen ein Wort $\widetilde{c} \in \Sigma^n$. Bei der Übertragung werde die Länge n des Codeworts nicht verändert, jedoch kann ein einzelnes Zeichen der Nachricht zufällig, aber mit gleichbleibender Wahrscheinlichkeit beeinflusst werden. Wir sagen bei der Übertragung von c sind t Fehler aufgetreten, falls der Hammingabstand $d(c, \widetilde{c}) := \{i \in n \mid c_i \neq \widetilde{c}_i\}$ gleich t ist. Bei der anschließenden Decodierung transformiert man das empfangene Codewort $\widetilde{c}$ in ein bezüglich der Hammingmetrik nächstliegendes Codewort $\overline{c} \in C$. Gibt es mehrere, so wählt man unter diesen eines zufällig. Dieses Decodierverfahren nennt man auch die Maximum–Likelihood–Decodierung, kurz MLD.

Dieser Abschnitt orientiert sich an [BBF+06], deshalb sei an dieser Stelle für weitere Erläuterungen und fehlende Beweise darauf verwiesen.

1.2.1. Lineare Codes

Wie wir bereits in der Einleitung bemerkt haben, werden wir in dieser Arbeit ausschließlich lineare Codes über endlichen Körpern betrachten.

1.2.1 Definition (Linearer Code)**.** Sei $\gamma : GF(q)^k \to GF(q)^n$ linear. Das Bild $C := \gamma(GF(q)^k)$ ist ein zu $GF(q)^k$ isomorpher Unterraum von $GF(q)^n$. C heißt dann ein linearer (n, k) – Code über $GF(q)$ oder kurz (n, k, q) – Code.

1.2.2 Definition (Generator- und Kontrollmatrix)**.** Die lineare Abbildung γ kann auch durch eine Multiplikation mit einer $k \times n$–Matrix Γ ausgedrückt werden:

$$\gamma : GF(q)^k \to GF(q)^n, v \mapsto v \cdot \Gamma.$$

Man erhält $\gamma(GF(q)^k) = C = \{v \cdot \Gamma \mid v \in GF(q)^k\}$. Dabei bilden die Zeilen von Γ eine Basis von C. Die darstellende Matrix Γ ist dabei im Allgemeinen nicht eindeutig.
Der Unterraum $C \leq GF(q)^n$ kann auch als Kern einer surjektiven, linearen Abbildung $\delta : GF(q)^n \to GF(q)^{n-k}$ formuliert werden. Jede $(n-k) \times n$–Matrix Δ vom Rang $n-k$ mit

$$C = \{w \in GF(q)^n \mid w \cdot \Delta^T = 0\} = \text{Kern}(\delta)$$

heißt Kontrollmatrix von C.

1.2.3 Definition. Ein linearer Code C heißt

- redundant, falls eine beliebige Generatormatrix Nullspalten enthält,

- projektiv, falls je zwei verschiedene Spalten einer Generatormatrix linear unabhängig sind.

1.2.4 Satz (Hammingmetrik)**.** *Die Funktion* $d : GF(q)^n \times GF(q)^n \to \mathbb{N}, (u, v) \mapsto d(u, v)$ *mit* $d(u, v) := |\{i \in n \mid u_i \neq v_i\}|$ *definiert eine Metrik, die Hammingmetrik.* $(GF(q)^n, d)$ *ist metrischer Raum und wird auch mit* $H(n, q)$ *bezeichnet.*
Für einen Vektor $v \in GF(q)^n$ *schreiben wir* $\text{gew}(v) := d(v, 0)$ *für dessen Hamminggewicht.*

Oft werden wir implizit die Existenz der Hammingmetrik voraussetzen und statt $H(n, q)$ nur $GF(q)^n$ schreiben.

1.2.5 Definition (Minimaldistanz)**.** Für einen linearen Code C sei mit $\text{dist}(C) := \min\{d(c, c') \mid c, c' \in C, c \neq c'\} = \min\{\text{gew}(c) \mid c \in C, c \neq 0\}$ seine Minimaldistanz bezeichnet.
Ein linearer Code C über $GF(q)$ mit $\text{dist}(C) = d$ heißt auch (n, k, d, q) – Code.

Ein (n, k, d, q) – Code erkennt $d-1$ Fehler und kann $\lfloor \frac{d-1}{2} \rfloor$ Fehler mit der Maximum–Likelihood–Decodierung korrigieren. Da $d \leq n$, folgt für die (n, k) – Codes C mit $n < 3$, dass diese nicht in der Lage sind, Fehler zu korrigieren.

1.2.6 Definition (Gewichtsverteilung)**.** Sei C ein beliebiger Code der Länge n und $A_i := \{c \in C | \, \text{gew}(c) = i\}$. x, y seien Unbestimmte über $\mathbb{C}$. Dann heißt das Polynom

$$A_C(x, y) = \sum_{i=0}^{n} A_i x^i y^{n-i} \in \mathbb{C}\,[x, y]$$

die Gewichtsverteilung von C.

1.2.7 Definition (Dualer Code)**.** Sei C ein (n, k, d, q) – Code. Mit Hilfe des Standardskalarprodukts $< .|. >: GF(q)^n \times GF(q)^n \to GF(q), (u, v) \mapsto \sum_{i=0}^{n} u_i v_i$ können wir einen weiteren Unterraum in $GF(q)^n$ identifizieren, den wir als den zu C dualen Code bezeichnen, kurz: $C^\perp$. Dabei bestehe $C^\perp$ aus allen Vektoren, die zu C orthogonal sind.

$$C^\perp := \{v \in GF(q)^n \mid \forall c \in C :< c|v >= 0\}.$$

1.2.8 Bemerkung. Es gilt $(C^\perp)^\perp = C$.

1.2.9 Satz. *Genau die Generatormatrizen von* C *sind die Kontrollmatrizen von* $C^\perp$*. Genauso sind genau die Kontrollmatrizen von* C *Generatormatrizen des dualen Codes.*

1.2.10 Satz (Eigenschaften der Kontrollmatrix)**.** *Jede Kontrollmatrix* Δ *eines linearen* (n, k, d, q) *– Codes* C *hat die folgenden Eigenschaften:*

- *Der Rang von* Δ *ist* $n - k$*,*

- *je $d-1$ Spalten von Δ sind linear unabhängig,*
- *es gibt d linear abhängige Spalten.*

Umgekehrt ist jede solche Matrix Kontrollmatrix eines (n,k,d,q) – Codes.

1.2.11 Hilfssatz. *Für jede Generatormatrix Γ und jede Kontrollmatrix Δ von C sind die Produkte $\Gamma \cdot \Delta^T$ bzw. $\Delta \cdot \Gamma^T$ Nullmatrizen. Umgekehrt ist auch jede $(n-k) \times n$–Matrix Δ mit vollem Zeilenrang und $\Gamma \cdot \Delta^T = 0_{k\times(n-k)}$ eine Kontrollmatrix von C.*

1.2.12 Definition (Informationsmenge)**.** Sei C ein (n,k,d,q) – Code und $I \subseteq n$ mit $|I| = k$, so dass jedes Codewort bereits durch Angabe der Werte an diesen k Koordinatenstellen eindeutig festgelegt ist. Solche Teilmengen I werden Informationsmengen genannt.

Der Rang jeder Generatormatrix von C ist k, daher gibt es mindestens eine Informationsmenge. Diese entspricht den Koordinatenpositionen von k linear unabhängigen Spaltenvektoren.

1.2.13 Bemerkung (Notation)*.* Seien a und b positive ganze Zahlen. Wir werden eine Matrix $A \in GF(q)^{a\times b}$ als Abbildung von $b = \{0,\ldots,b-1\}$ nach $GF(q)^a$ auffassen. Insofern schreiben wir auch für die i–te Spalte $A(i)^T$. Das Transponieren ist nötig, da wir Vektoren – wie in der Codierungstheorie üblich – in Zeilenform notieren werden.

1.2.14 Definition (Punktierung)**.** Sei C ein (n,k,d,q) – Code mit $n > k$ und $i \in n$. Weiter sei $\Gamma = \big(\Gamma(0)^T|\ldots|\Gamma(n-1)^T\big)$ eine Generatormatrix von C. Dann bezeichnen wir mit C_i den Code mit Generatormatrix $\widetilde{\Gamma} = \big(\Gamma(0)^T|\ldots|\Gamma(i-1)^T|\Gamma(i+1)^T|\ldots|\Gamma(n-1)^T\big)$. Wir sagen C_i geht aus C durch Punktieren an der Position i hervor.

Existiert eine Informationsmenge I, die i nicht enthält, so ist der Code C_i ein $(n-1,k,\geq d-1,q)$ – Code. Die Definition ist in dieser Hinsicht leicht abweichend von [BBF$^+$06], da dies um den Parameter k zu erhalten dort als Voraussetzung zur Punktierung gefordert wird.
Als Verallgemeinerung definieren wir für $J \subseteq n$ die Generatormatrix des in J punktierten Code C_J durch Streichen aller Spalten $j \in J$ von Γ. Der Code C_J ist dann ein k'–dimensionaler Unterraum, $k' \leq k$, von $GF(q)^{n-|J|}$.

1.2.15 Definition (Verkürzung)**.** Sei C ein linearer (n,k,d,q) – Code und $J \subseteq n$. Wir definieren einen linearen Code $\widetilde{C}$ wie folgt:

$$\widetilde{C} := \big\{c \in C \mid \forall j \in J : c_j = 0\big\}.$$

Der lineare Code $C_{\setminus J} := \widetilde{C}_J$ heißt der in J verkürzte Code.

Der Code $\widetilde{C}$ besteht aus allen Codevektoren, die an den Koordinatenpositionen $j \in J$ gleich Null sind. Da diese Stellen dann keinerlei Information mehr tragen, können sie auch, ohne Abstriche an die fehlerkorrigierende Eigenschaft in Kauf nehmen zu müssen, gestrichen werden. Wir erhalten hierdurch den Code $C_{\setminus J}$.

Ist $k > 1$, $J = \{i\}$ und $\Gamma(i)^T$ keine Nullspalte für eine Generatormatrix Γ, so ist $C_{\setminus i} := C_{\setminus\{i\}}$ ein $(n-1, k-1, \geq d, q)$ – Code.

1.2.2. Isometrien

1.2.16 Definition (Isometrien)**.** Funktionen $\iota : H(n,q) \to H(n,q)$ mit

$$d(u,v) = d(\iota(u), \iota(v)) \quad \forall u, v \in H(n,q)$$

heißen Isometrien.

1.2.17 Definition. Die linearen (n,k) – Codes C und C' heißen isometrisch, falls es eine Isometrie ι gibt mit $\iota(C) = C'$.

1.2.18 Definition. Wir werden die folgenden Spezialfälle von Isometrien in dieser Arbeit näher betrachten. Dabei sind die zuerst genannten jeweils Sonderfälle der späteren.

1. Permutationsisometrien

   Abbildungen $\iota : H(n,q) \to H(n,q), (v_0, \ldots, v_{n-1}) \mapsto (v_{\pi^{-1}(0)}, \ldots, v_{\pi^{-1}(n-1)})$ für ein $\pi \in S_n$ sind offensichtlich Isometrien und werden Permutationsisometrien genannt. Codes, die durch solche Isometrien aufeinander übergeführt werden können, heißen entsprechend permutationsisometrisch.

2. Lineare Isometrien

   Isometrien, die gleichzeitig lineare Abbildungen sind, werden lineare Isometrien genannt, die Codes linear isometrisch.

3. Semilineare Isometrien

   Eine Abbildung $\sigma : GF(q)^n \to GF(q)^n$ heißt semilinear, falls es einen Körperautomorphismus α gibt, so dass für alle $u, v \in GF(q)^n$ und für alle $\kappa \in GF(q)$ die folgenden Eigenschaften erfüllt sind:

   $$\sigma(u+v) = \sigma(u) + \sigma(v)$$
   $$\sigma(\kappa v) = \alpha(\kappa)\sigma(v)$$

   Codes, die mit Hilfe semilinearer Isometrien aufeinander abgebildet werden können, heißen semilinear isometrisch.

Semilineare Isometrien, die einen Code C auf sich selbst abbilden, nennen wir Automorphismen von C, entsprechend lineare Isometrien mit dieser Eigenschaft lineare Automorphismen und Permutationsisometrien Permutationsautomorphismen. Die Menge aller Automorphismen von C bezeichnen wir mit $Aut(C)$, die Menge der linearen mit $LAut(C)$ und diejenigen die Koordinatenvertauschungen entsprechen $PAut(C)$. Es gilt:

$$PAut(C) \leq LAut(C) \leq Aut(C) \leq Aut(GF(q)^n).$$

Da lineare Isometrien das Hamminggewicht erhalten ($\iota(0) = 0$), wird jeder Einheitsvektor $e^{(i)}$ auf ein echtes Vielfaches (nicht Null) eines Einheitsvektors abgebildet. Da die Summe zweier verschiedener Einheitsvektoren vom Hamminggewicht 2 ist, muss auch das Bild dieser Summe vom Gewicht 2 sein. Also werden die gewählten Einheitsvektoren auf Vielfache verschiedener Einheitsvektoren abgebildet. Somit lässt sich die lineare Isometrie ι eindeutig durch die Angabe einer Permutation $\pi \in S_n$ und einer Abbildung $\varphi : n \to GF(q)^*$ beschreiben:

$$\iota\big(e^{(i)}\big) = \varphi\big(\pi(i)\big)e^{(\pi(i))}.$$

Andererseits ist auch jedes solche Paar $(\pi; \varphi)$ eine lineare Isometrie.

Die lineare Isometrie ι lässt sich auch in Matrixform durch Angabe der Bilder der Einheitsvektoren notieren. Ist $\iota = (\pi; \varphi)$ und $j = \pi(i)$, so ist die i–te Spalte der zugehörigen Matrixdarstellung $M_{(\pi;\varphi)} \in GL_n(q)$ von folgender Form:

$$\big(M_{(\pi;\varphi)}(i)\big)^T = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \varphi(j) \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow j = \pi(i)$$

Diese Untergruppe der invertierbaren Matrizen $GL_n(q)$, die den linearen Isometrien entsprechen, nennt man auch die Gruppe der monomialen Matrizen $M_n(q)$.

Es ergibt sich für $v \in GF(q)^n$:

$$\iota(v) = (\pi; \varphi)\big((v_0, \ldots, v_{n-1})\big) = \big(\varphi(0)v_{\pi^{-1}(0)}, \ldots, \varphi(n-1)v_{\pi^{-1}(n-1)}\big) = v \cdot M^T_{(\pi;\varphi)}.$$

1.2.19 Hilfssatz. *Die Gruppe der linearen Isometrien von $H(n, q)$ ist isomorph zu dem Kranzprodukt $GF(q)^* \wr_n S_n$:*

$$M_n(q) \simeq GF(q)^* \wr_n S_n.$$

1.2.20 Definition. Sei $q = p^r$ und p prim. Die Abbildung

$$\tau : GF(q) \to GF(q), \mu \mapsto \mu^p$$

ist ein Körperautomorphismus und wird Frobenius Automorphismus von $GF(q)$ über $GF(p)$ genannt.

1.2.21 Hilfssatz. *Die Gruppe der Körperautomorphismen von $GF(q)$, auch Galoisgruppe $Gal(q)$ von $GF(q)$ über $GF(p)$ genannt, ist zyklisch und wird durch den Frobenius Automorphismus erzeugt. Insbesondere ist die Ordnung $|Gal(q)| = r$.*

1.2.22 Bemerkung. Wir reservieren den Buchstaben τ für den Frobenius Automorphismus, sowie – falls aus dem Zusammenhang nicht eine weitere Bedeutung hervorgeht – die Buchstaben p, q, r zur Beschreibung des Körpers. Jeder Körperautomorphismus α ist somit durch eine eindeutige Potenz $x \in \{0, \ldots, r-1\} : \tau^x = \alpha$ beschreibbar

1.2.23 Folgerung. *Die Untergruppen $U \leq Gal(q)$ sind zyklisch und werden durch die minimale Potenz $x \in \{0, \ldots, r-1\} : \tau^x \in U$ des Frobenius Automorphismus erzeugt.*

1.2.24 Satz. *Für $n \geq 3$ sind genau die Abbildungen der Form $(\varphi; (\alpha, \pi))$, wobei $(\varphi; \pi)$ eine lineare Isometrie sei und α ein Körperautomorphismus, Isometrien, welche Unterräume auf Unterräume abbilden. Diese Menge bildet eine Gruppe*

$$GF(q)^{*n} \rtimes_\theta (Gal(q) \times S_n),$$

mit

$$\begin{aligned} \theta(Gal(q) \times S_n) &\to Aut(GF(q)^{*n}) \\ (\alpha, \pi) &\mapsto \theta(\alpha, \pi) \end{aligned}$$

und

$$\theta(\alpha, \pi)(\varphi)(i) := \varphi_{(\alpha,\pi)}(i) := \alpha(\varphi(\pi^{-1}(i))), i \in n,$$

die Gruppe der semilinearen Isometrien. Diese operiert auf der Menge $GF(q)^n$ wie folgt:

$$\begin{aligned} \left(GF(q)^{*n} \rtimes_\theta (Gal(q) \times S_n)\right) \times GF(q)^n &\to GF(q)^n \\ \left((\varphi; (\alpha, \pi)), v\right) &\mapsto \alpha(v) \cdot M^T_{(\varphi;\pi)} \end{aligned}$$

Beweis. Siehe [BBF+06]. □

1.2.25 Definition. Wir nennen einen linearen (n, k) – Code C zerlegbar, falls er linear isometrisch zu einem Code C' mit Generatormatrix

$$\Gamma := \begin{pmatrix} \Gamma_1 & 0 \\ 0 & \Gamma_2 \end{pmatrix} =: \Gamma_1 \dot{+} \Gamma_2$$

ist. Wir schreiben C' auch als $C' := C(\Gamma_1)\dot{+}C(\Gamma_2)$. In dem anderen Fall sprechen wir von unzerlegbaren Codes beziehungsweise unzerlegbaren Generatormatrizen.

1.2.26 Satz. *Ist $k < n$ und C ein beliebiger (n,k) – Code mit $dist(C) = d$. Dann gibt es einen unzerlegbaren (n,k) – Code C' mit $dist(C') \geq d$.*

1.2.27 Satz. *Es seien C, C' unzerlegbare, isometrische (n,k) – Codes. Dann gibt es auch eine semilineare Isometrie $\big(\varphi;(\alpha,\pi)\big) \in GF(q)^{*n} \rtimes_\theta (Gal(q) \times S_n)$, so dass $\alpha(C) \cdot M^T_{(\varphi;\pi)} = C'$.*

Beweis. Siehe [Mon87]. □

Diese Aussage gilt nicht für allgemeine (n,k) – Codes. Hier gibt es Beispiele, bei denen zwei zerlegbare Codes, die nicht semilinear isometrisch sind, durch eine Isometrie aufeinander übergeführt werden können.

1.2.28 Folgerung. *Wir sind also, mit der Bestimmung eindeutiger Repräsentanten der semilinearen Isometrieklassen linearer Codes, auch in der Lage für den codierungstheoretisch wichtigen Fall der unzerlegbaren Codes eindeutige Repräsentanten der Isometrieklassen zu berechnen.*

2. Kanonisierung diskreter Strukturen

Wir haben in Kapitel 1 bereits gesehen, dass wir sowohl die linearen als auch die semilinearen Isometrieklassen linearer Codes als Bahnen einer Gruppenoperation auffassen können. Um einen eindeutigen Repräsentanten, wir nennen diesen auch kanonisches Element, einer Bahn effizient berechnen zu können, entwickeln wir in diesem Kapitel ein allgemeines Kanonisierungsverfahren für Gruppenoperationen. Die Entwicklung des Verfahrens erfolgt nach [Gug05], ist jedoch an einigen Stellen modifiziert.

2.1. Kanonisierer und kanonische Transversalen

2.1.1 Definition. Wir bezeichnen zu einer Gruppe G die Menge aller Untergruppen von G mit

$$\mathcal{L}(G) := \{U \mid U \leq G\}.$$

2.1.2 Definition. Ein System kanonischer Transversalen auf der G–Menge X sei eine Abbildung

$$Can : \mathcal{L}(G) \to 2^X,$$

so dass für alle $H \leq G$ gilt:

$Can(H)$ ist H–Transversale von X.

Wir nennen $Can(H)$ die H–kanonische Transversale und ein $x \in Can(H)$ einen H–kanonischen Repräsentanten bzgl. Can.

2.1.3 Bemerkung. Wir fordern nicht – im Gegensatz zu [Gug05] – die Inklusionseigenschaft

$$H_1 \leq H_2 \leq G \Rightarrow Can(H_1) \supseteq Can(H_2).$$

Diese ist zum Einen für die Entwicklung des Verfahrens nicht notwendig, andererseits bei der Operation des Hebens eines Kanonisierers vgl. Satz 2.3.1 auch nicht für das resultierende System kanonischer Transversalen zu garantieren.

2.1.4 Definition. Es sei Can ein System kanonischer Transversalen. Wir nennen eine Abbildung (Algorithmus)

$$\begin{aligned} can : \mathcal{L}(G) \times X &\to G \times \mathcal{L}(G) \\ (H, x) &\mapsto (\gamma(x), H_x) \end{aligned}$$

Kanonisierer bzgl. Can, falls $\gamma(x)x \in Can(H)$ für alle $x \in X$ gilt, und H_x der Stabilisator von x in H ist.

2.1.5 Beispiel. Nehmen wir weiterhin an, dass durch $(X, \leq)$ eine Totalordnung auf X gegeben sei. Dann können wir als System kanonischer Transversalen die Minima der Bahnen wählen:

$$Can_{min}(H) := \{x \in X \mid \forall h \in H : x \leq hx\}.$$

Durch Anwenden aller Gruppenelemente $h \in H$ auf die Eingabe $x \in X$ und paarweisen Vergleich der Elemente hx, haben wir bereits einen Kanonisierer bzgl. Can_{min} entwickelt. Jedoch ist dieser Algorithmus offensichtlich in der Anwendung nur für sehr kleine Gruppenordnungen geeignet.

Um einen vertretbaren Rechenaufwand zu garantieren, müssen möglichst viele Gruppenelemente von der Betrachtung ausgeschlossen werden. Dazu gibt es eine Vielzahl von Möglichkeiten, von denen einige hier erwähnt werden sollen. Weitere finden sich in [Leo84] oder [Gug05]. Grundsätzlich bieten sich zwei Vorgehensweisen zur Minimierung der Anzahl der betrachteten Gruppenelementen während der Berechnung von $can(H, x)$:

- Ausnutzung des Stabilisators H_x,
- Ausgrenzen von Teilmengen, in denen ein Auffinden eines kanonisierenden Elements ausgeschlossen werden kann.

2.2. Anwendung des Homomorphieprinzips

Wie wir bereits in Kapitel 1 gesehen haben, kann man mit Hilfe eines G–Homomorphismus $f : X \to Y$ eine Transversale von $G \backslash\backslash X$ aus einer gegebenen Transversalen $T_{G \backslash\backslash Y}$ berechnen:

$$T_{G \backslash\backslash X} := \bigcup_{y \in T_{G \backslash\backslash Y}} T_{G_y \backslash\backslash f^{-1}(y)}.$$

Wir wollen diese Tatsache nutzen um aus gegebenen Systemen kanonischer Transversalen mit entsprechenden Kanonisierern neue Systeme mit effizienteren Kanonisierern zu definieren.

2.2.1 Satz. *Es seien X und Y G–Mengen, sowie Can_X und Can_Y Systeme kanonischer Transversalen mit Kanonisierern can_X und can_Y. Weiterhin sei $f : X \to Y$ ein G–Homomorphismus. Durch*

$$Can_f : \mathcal{L}(G) \to 2^X$$
$$H \mapsto Can_f(H) := \{x \in X \mid f(x) \in Can_Y(H) \wedge x \in Can_X(H_{f(x)})\}$$

ist ein weiteres System kanonischer Transversalen auf X definiert. Der Algorithmus 2.1 ist ein Kanonisierer bzgl. Can_f:

Algorithmus 2.1 $can_f(H, x)$ – Kanonisierer auf Basis des Homomorphieprinzips

Input: $(H, x) \in \mathcal{L}(G) \times X$
Output: $(\gamma(x), H_x)$ mit $\gamma(x)x \in Can_f(H)$
$(g, H_{f(x)}) \leftarrow can_Y(H, f(x))$;
$(h, H_{gx}) \leftarrow can_X\big(gH_{f(x)}g^{-1}, gx\big)$;
return $(hg, g^{-1}H_{gx}g)$;

Beweis. Für jedes $H \leq G$ ist f auch ein H–Homomorphismus, somit ist $Can_f(H)$ nach dem Homomorphieprinzip für Gruppenoperationen eine Transversale von $H\backslash\backslash X$. Für den angegebenen Algorithmus genügt es zu bemerken, dass $H_{gx} = gH_xg^{-1}$ und $H_{gx} \leq H_{f(gx)} = gH_{f(x)}g^{-1}$ für alle $g \in H, x \in X$ gilt. □

2.2.2 Bemerkung. Erfüllen Can_X und Can_Y die Inklusionseigenschaft, so auch Can_f.

Beweis. Seien $H_1 \leq H_2 \leq G$ und $x \in Can_f(H_2)$ beliebig. Dann folgt aus der Inklusionseigenschaft von Can_Y, dass $f(x)$ ebenfalls in $Can_Y(H_1)$ liegt. Der Stabilisator $(H_1)_{f(x)}$ ist eine Untergruppe von $(H_2)_{f(x)}$ und somit gilt $x \in Can_X((H_1)_{f(x)})$. Damit gilt aber auch $x \in Can_f(H_1)$ und schließlich $Can_f(H_1) \supseteq Can_f(H_2)$. □

2.2.3 Bemerkung (Geeignete Homomorphismen). Die Kanonisierung über can_f ist dann vorteilhafter gegenüber der direkten Kanonisierung mittels can_X, falls entweder

- die Kanonisierung mittels can_Y sehr einfach geschehen kann, z.B. die Sortierung eines Vektors von Zahlen, oder
- die Gruppen folgendermaßen ineinander liegen: $H > H_{f(x)} > H_x$. In diesem Falle ergeben sich als Bahnenordnungen $\frac{|H|}{|H_{f(x)}|}$ bzw. $\frac{|H_{f(x)}|}{|H_x|}$. Unter der Annahme, dass der Stabilisator während der Kanonisierung weitestgehend ausgenutzt werden kann und damit die Kanonisierung mittels can_X bzw. can_Y im Wesentlichen von der Ordnung der entsprechenden Bahnen abhängen, erhalten wir eine Aufwandsabschätzung in Abhängigkeit von $\frac{|H|}{|H_{f(x)}|} + \frac{|H_{f(x)}|}{|H_x|}$ im Gegensatz zu $\frac{|H|}{|H_x|} = \frac{|H|}{|H_{f(x)}|} \cdot \frac{|H_{f(x)}|}{|H_x|}$ bei der direkten Kanonisierung mittels can_X.

2.2.4 Hilfssatz. *Es seien X, Y zwei beliebige G–Mengen, damit ist auch das direkte Produkt $X \times Y$ eine G–Menge. Weiterhin seien Can_X bzw. Can_Y entsprechende Systeme kanonischer Transversalen auf X bzw. Y mit Kanonisierern can_X und can_Y. Durch*

$$Can_{X\times Y} : \mathcal{L}(G) \to 2^{X\times Y}$$
$$H \mapsto \{(x,y) \in X \times Y \mid x \in Can_X(H) \wedge y \in Can_Y(H_x)\}$$

ist ein System kanonischer Transversalen auf $X \times Y$ definiert. Der Algorithmus 2.2 ist ein Kanonisierer bzgl. $Can_{X\times Y}$:

Algorithmus 2.2 $can_{X\times Y}(H,(x,y))$ – Kanonisierer für das direkte Produkt von G–Mengen

Input: $(H,(x,y)) \in \mathcal{L}(G) \times (X \times Y)$
Output: $(\gamma(x,y), H_{(x,y)})$ mit $\gamma((x,y))(x,y) \in Can_{X\times Y}(H)$
  $(g, H_x) \leftarrow can_X(H,x)$;
  $(h, H_{(gx,gy)}) \leftarrow can_Y(gH_xg^{-1}, gy)$;
  return $(hg, g^{-1}H_{(gx,gy)}g)$;

Beweis. Wir benutzen Algorithmus 2.1 und wählen als G–Homomorphismus die Abbildung

$$f : X \times Y \to X, (x,y) \mapsto x.$$

Für die Kanonisierung $can_{X\times Y}(gH_xg^{-1}, g(x,y))$ können wir dann auch can_Y wählen, da die erste Komponente gx des Elements $g(x,y)$ durch die Gruppe gH_xg^{-1} fix bleibt. □

2.2.5 Folgerung. *Es sei eine Folge von G–Mengen $(X_i)_{i\in n}$ mit Systemen von kanonischen Transversalen Can_{X_i} und zugehörigen Kanonisierern can_{X_i} für jedes $i \in n$ gegeben. Dann ist mittels*

$$Can_{\times_{i\in n} X_i}(H) := \{(x_i)_{i\in n} \in \bigtimes_{i\in n} X_i \mid \forall i \in n : x_i \in Can_{X_i}(H \cap \bigcap_{j<i} H_{x_j})\}$$

ein System kanonischer Transversalen auf der G–Menge $\times_{i\in n} X_i$ definiert. Der Algorithmus 2.3 ist ein Kanonisierer bzgl. $Can_{\times_{i\in n} X_i}$.

Beweis. Induktion nach n. □

Algorithmus 2.3 $can_{\times_{i\in n} X_i}(H,(x_i)_{i\in n})$ – Kanonisierer für eine Folge von G–Mengen

Input: $(H,(x_i)_{i\in n}) \in \mathcal{L}(G) \times \times_{i\in n} X_i$
Output: $(\gamma((x_i)_{i\in n}), H_{(x_i)_{i\in n}})$ mit $\gamma((x_i)_{i\in n})(x_i)_{i\in n} \in Can_{\times_{i\in n} X_i}(H)$

```
g ← id;
E ← H;
for i ← 0 to n − 1 do
    (h, E) ← can_{X_i}(E, gx);
    E ← hEh^{-1};
    g ← hg;
end for
return (g, g^{-1}Eg);
```

2.2.6 Folgerung. *Es seien X und $Y_i, i \in n$ G–Mengen mit entsprechenden Systemen kanonischer Transversalen Can_X bzw. $Can_{Y_i}, i \in n$ und zugehörigen Kanonisierern can_X und $can_{Y_i}, i \in n$. Weiter sei eine Folge von G–Homomorphismen $f_i : X \to Y_i$ gegeben. Dann ist auch mittels*

$$Can_{(f_i)_{i\in n}}(H) := \{x \in X \mid (f_i(x))_{i\in n} \in Can_{\times_{i\in n} Y_i}(H) \wedge x \in Can_X(H_{(f_i(x))_{i\in n}})\}$$

ein System kanonischer Transversalen $Can_{(f_i)_{i\in n}}$ auf X definiert. Der Algorithmus 2.4 ist ein Kanonisierer zu diesem System.

Beweis. Die Folge $(f_i)_{i\in n}$ der G–Homomorphismen ergibt den G–Homomorphismus

$$\overline{f} : X \to \bigtimes_{i\in n} Y_i, \quad x \mapsto (f_i(x))_{i\in n}.$$

Wir wenden nun auf $\overline{f}$ die vorausgegangenen Sätze an. □

2.2.7 Bemerkung. Für alle oben genannten Systeme kanonischer Transversalen vererbt sich die Inklusionseigenschaft.

2.2.8 Bemerkung. Eine erhebliche Aufwandsreduktion erhält man analog zu oben durch günstige Verteilung der Indizes der Untergruppenkette

$$G \geq G_{f_0(x)} \geq G_{(f_0(x),f_1(x))} \geq \ldots \geq G_{(f_i(x))_{i\in n}} \geq G_x$$

oder möglichst einfache Mengen Y_i mit sehr effizienten Kanonisierern.

Mit Hilfe von G–Homomorphismen erreichen wir also, dass der algorithmisch teure Aufruf des Kanonisierers can_X nur noch für die möglichst kleine Untergruppe $H_{(f_i(x))_{i\in n}}$ zu erfolgen hat.

Algorithmus 2.4 $can_{(f_i)_{i\in n}}(H,x)$ – Kanonisierung mittels mehrfacher Anwendung des Homomorphieprinzips

Input: $(H,x) \in \mathcal{L}(G) \times X$
Output: $(\gamma(x), H_x)$ mit $\gamma(x)x \in Can_{(f_i)_{i\in n}}(H)$

```
g ← id;
E ← H;
for i ← 0 to n − 1 do
    // Kanonisierung mittels can_{×_{i∈n} Y_i}
    (h, E) ← can_{Y_i}(E, f_i(gx));
    E ← hEh^{-1};
    g ← hg;
end for
(h, E) ← can_X(E, gx);
E ← hEh^{-1};
g ← hg;
return (g, g^{-1}Eg);
```

2.3. Kanonisierung entlang von Untergruppenketten

Wir möchten nun noch weiter gehen und die Transversale Can_X sowie den Kanonisierer can_X möglichst geschickt wählen. Wir geben dazu eine Idee an, wie wir aus einem System kanonischer Transversalen Can einer Untergruppe $G' \leq G$ ein System kanonischer Transversalen $Can^{\uparrow G}$ für die G–Menge X erhalten. Vorteil dieses sogenannten Hebens eines G'–Kanonisierers zu einem G–Kanonisierers ist, dass auch der G'–Kanonisierer auf G'–Homomorphismen zur Effizienzsteigerung zurückgreifen kann. Wendet man dieses Heben zum Beispiel iterativ auf eine Stabilisatorkette an, so wird sich zeigen, dass die Festlegung der Bilder einiger Basiselemente erst die Definition von geeigneten Homomorphismen erlaubt. Die operierende Untergruppe ist dementsprechend der punktweise Stabilisator dieser Basiselemente.

2.3.1 Satz (Heben eines Kanonisierers einer Untergruppe zu einem Kanonisierer der operierenden Gruppe)**.** *Es sei $G' \leq G$ und X eine mittels „$\leq$“ totalgeordnete G–Menge. Die Abbildung $Can : \mathcal{L}(G') \to 2^X$ sei ein System kanonischer Transversalen für die Operation von G' auf X. Wir erhalten durch die Definition*

$$Can^{\uparrow G}(H) := \left\{x \in X \mid x \in Can(H \cap G') \wedge \forall h \in H, hx \in Can(H \cap G') : x \leq hx\right\}$$

ein System kanonischer Transversalen $Can^{\uparrow G}$ für die Operation von G auf X. Der Algorithmus 2.5 ist ein Kanonisierer zu $Can^{\uparrow G}$.

Beweis. Wir zeigen zunächst, dass $Can^{\uparrow G}(H)$ eine Transversale von $H\backslash\backslash X$ ist. Seien dazu $x_1, x_2 \in Can^{\uparrow G}(H), x_1 \neq x_2$ beliebig und ohne Beschränkung der Allgemeinheit

Algorithmus 2.5 $can^{\uparrow G}(H, x)$ – Heben eines G'–Kanonisierers zu einem G–Kanonisierer für $G' < G$

Input: $(H, x) \in \mathcal{L}(G) \times X$
Output: $(\gamma(x), H_x)$ mit $\gamma(x)x \in Can^{\uparrow G}(H)$

```
g ← NIL;
T ← Transversale von (H ∩ G')\H;
for all t ∈ T do
    (h_t, E_t) ← can(H ∩ G', tx);
    if g ≠ NIL ∧ h_t tx = gx then
        E ← E ∪ {g^{-1} h_t t};
    end if
    if g = NIL ∨ h_t tx < gx then
        g ← h_t t;
        E ← E ∪ t^{-1} E_t t;
    end if
end for
return (g, ⟨E⟩);
```

$x_1 < x_2$. Angenommen es existiere ein $h \in H$ sd. $hx_2 = x_1$, dann folgt:

$$x_1, x_2 \in Can^{\uparrow G}(H) \Rightarrow x_1, x_2 \in Can(H \cap G') \wedge x_2 \leq hx_2 = x_1 < x_2. \text{ Widerspruch.}$$

Andererseits ist diese Menge auch vollständig, da es zu jedem $x \in X$ ein $(H \cap G')$–kanonisches Element $h'x$ gibt. Zu diesem Element gibt es ein $\widetilde{x} \in Can^{\uparrow G}(H)$, das ebenfalls $(H \cap G')$–kanonisch ist und $h\widetilde{x} = h'x$ für ein $h \in H$ gilt. Also sind $\widetilde{x}$ und x H–abhängig. Jede H–Bahn wird also in der Menge $Can^{\uparrow G}(H)$ repräsentiert.

Es bleibt zu zeigen, dass die Ausgabe des Kanonisierers richtig ist. Dazu sei $g_{opt} = h_{opt}t_{opt}$ das Gruppenelement, welches am Ende des Algorithmus ausgegeben wird. Offensichtlich ist $g_{opt}x = h_{opt}t_{opt}x \in Can(H \cap G')$. Wir zeigen, dass das Bahnenelement auch minimal unter allen $(H \cap G')$–kanonischen Bahnenelementen ist. Sei dazu $h \in H$ beliebig mit $hx \in Can(H \cap G')$. Zu h existiert ein eindeutiges Element $t \in T$ und $\overline{h} \in H \cap G'$, so dass $h = \overline{h}t$ gilt. Somit liegen tx und hx in der gleichen $(H \cap G')$–Bahn, insbesondere gilt also $h_t tx = hx$. Da $g_{opt}x$ unter allen $h_s sx$, $s \in T$ minimal ist, folgt $g_{opt}x \leq hx$.

Abschließend bleibt zu zeigen, dass der Kanonisierer auch die Automorphismengruppe H_x richtig bestimmt. Es werden jeweils nur Automorphismen von x zu der Menge E hinzugefügt. Damit ist $\langle E \rangle \leq H_x$, wir zeigen nun auch die Umkehrung. Sei also $a \in H_x$ beliebig. Für das Gruppenelement $g_{opt}a$ existiert wiederum eine Zerlegung mit $t \in T, h \in H \cap G'$ sd. $g_{opt}a = ht$. Da $htx = g_{opt}ax = g_{opt}x \in Can(H \cap G')$ gilt, folgt hieraus wegen den Eigenschaften von Can und der $(H \cap G')$–Abhängigkeit von tx und htx:

$$g_{opt}x = htx = h_t tx \in Can(H \cap G').$$

Nach der Definition des Algorithmus, muss also $g_{opt}^{-1}h_t t$ in E liegen. Es gilt weiter:

$$\begin{aligned}
&h_{opt}t_{opt}x = g_{opt}x = htx = ht(t^{-1}h_t^{-1}g_{opt}x) = hh_t^{-1}g_{opt}x = hh_t^{-1}h_{opt}t_{opt}x\\
&\Rightarrow h_{opt}^{-1}hh_t^{-1}h_{opt} \in H_{t_{opt}x} = E_{t_{opt}}\\
&\Rightarrow g_{opt}^{-1}hh_t^{-1}g_{opt} = t_{opt}^{-1}h_{opt}^{-1}hh_t^{-1}h_{opt}t_{opt} \in E.
\end{aligned}$$

Wir erhalten somit:

$$a = g_{opt}^{-1}ht = \underbrace{g_{opt}^{-1}hh_t^{-1}g_{opt}}_{\in E}\underbrace{g_{opt}^{-1}h_t t}_{\in E} \in \langle E \rangle.$$

□

2.3.2 Bemerkung. Beim Heben eines G'–Kanonisierers zu einem G–Kanonisierer werden die Eigenschaften des Stabilisators H_x nur unzureichend ausgenutzt. Aus dem Beweis des Satzes ergibt sich, dass in dem vorgestellten Algorithmus 2.5 bei jedem Ersetzen von g_{opt} durch einen neueren, optimalen Kandidaten auch die Menge E nur mit $E \leftarrow t^{-1}E_t t$ neu gesetzt werden könnte. Wir werden uns im nachfolgenden Abschnitt dieser Tatsache näher widmen und ein Verfahren angeben, wie wir die bekannten Automorphismen E besser einfließen lassen können.

2.3.3 Folgerung. *Es sei die Gruppe G mit einer Untergruppenkette $G = G^{(0)} \geq \ldots \geq G^{(r)} = \{\mathrm{id}\}$ gegeben. Weiter sei $Can_{id} : \mathcal{L}(\{\mathrm{id}\}) \to 2^X, \{\mathrm{id}\} \mapsto X$ das triviale System kanonischer Transversalen zu der Gruppe $G^{(r)}$. Es gilt:*

$$Can_{min} = (\cdots((Can_{id})^{\uparrow G^{(r-1)}})\cdots)^{\uparrow G^{(0)}}.$$

2.3.4 Hilfssatz. *Wir können den Kanonisierer entlang einer Untergruppenkette auch als Backtrackvorschrift formulieren. Dazu sei H eine Untergruppe von G und $H^{(i)} := H \cap G^{(i)}, i \leq r$. Weiter sei $T^{(i)}$ eine Rechtstransversale von $H^{(i+1)} \backslash H^{(i)}, i < r$. Außerdem seien für die Ebenen $i < r$ mögliche Testfunktion*

$$gx <_i hx \Rightarrow \forall h_1, h_2 \in H^{(i+1)} : h_1 gx < h_2 gx$$

zum frühzeitigen Abschneiden von Teilbäumen, die problemspezifisch bereit gestellt werden können, gegeben. Der Test auf Ebene r sei durch eine beliebige Totalordnung $(X, \leq)$ gegeben:

$$x \leq_r y :\Longleftrightarrow x \leq y.$$

Der Algorithmus 2.6 auf der nächsten Seite ist ein Kanonisierer zu Can_{min}.

Algorithmus 2.6 can_{min}– Kanonisierung entlang einer Untergruppenkette als Backtrack Verfahren

Input: $x \in X$ eine diskrete Struktur
Input: $\{T^{(0)}, \ldots, T^{(r-1)}\}$ Rechtstransversalen der Untergruppenkette
Output: $g \in H$ kanonisierendes Element
Output: $\langle S \rangle = H_x$

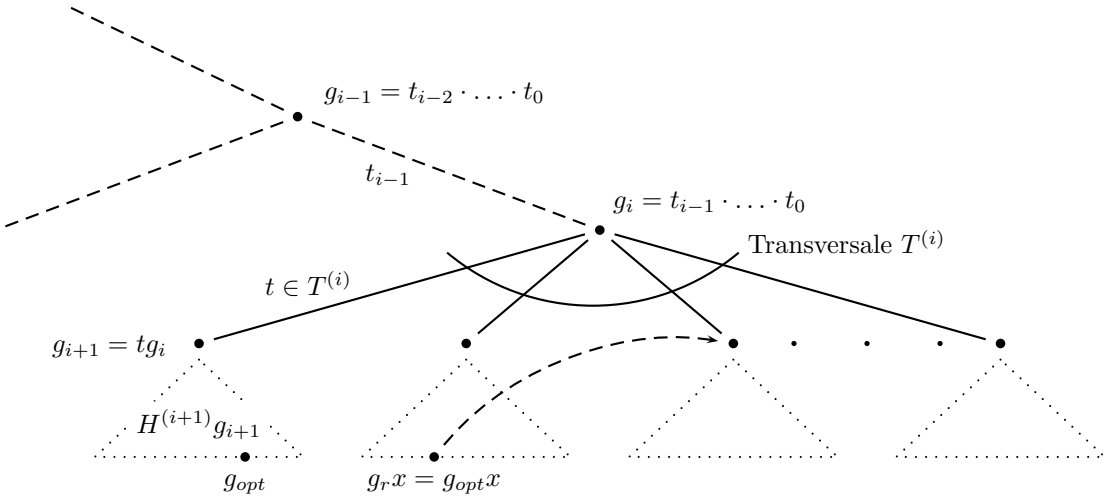
1: $g_0 \leftarrow \mathrm{id}$;
2: $g_{opt} \leftarrow \mathrm{id}$;
3: $S \leftarrow \emptyset$;
4: **while** Durchlaufe Backtrackbaum ohne Wurzel mittels Rechtstransversalenergänzung **do**
5: $i \leftarrow$ Ebene des aktuellen Knotens;
6: $j \leftarrow$ Index der eingehenden Kante;
7: $g_i \leftarrow T^{(i-1)}[j] \cdot g_{i-1}$;
8: **if** $g_i x >_i g_{opt} x$ **then**
9: **continue on level** i; // Überspringe darunterliegenden Teilbaum
10: **end if**
11: **if** $i = r \wedge g_i x = g_{opt} x \wedge g_i \neq g_{opt}$ **then**
12: $S \leftarrow S \cup \{g_{opt}^{-1} g_i\}$;
13: $m \leftarrow \max\{m' \leq r \mid g_i g_{opt}^{-1} \in G^{(m')}\}$;
14: **continue on level** $m + 1$;
15: **end if**
16: **if** $g_i x <_i g_{opt} x$ **then**
17: $g_{opt} \leftarrow g_i$;
18: **end if**
19: **if** $i < r$ **then**
20: **continue on level** $i + 1$;
21: **else**
22: **continue on level** i;
23: **end if**
24: **end while**
25: **return** (g_{opt}, S);

Beweis. Der Algorithmus liefert die gewünschte Ausgabe auf Grund der Folgerung 2.3.3, falls das Vorgehen in der IF-Bedingung ab Zeile 11 gerechtfertigt ist. Es sei also ein Gruppenelement $g_r \neq g_{opt}$ mit $g_r x = g_{opt} x$ erreicht. Weiter seien $g_{opt} = t_{r-1} \cdots t_0$ und $g_r = s_{r-1} \cdots s_0$ die Darstellungen mittels der Rechtstransversalenelemente $t_i, s_i \in T^{(i)}$, $i \in r$ und m wie im Algorithmus gewählt. Es gilt also:

$$t_l = s_l \text{ für } l < m \text{ und } t_m \neq s_m.$$

Übertragen wir die Situation wieder in die des Hebens eines G'–Kanonisierers, so ist also $G' = G^{(m+1)}$ und $G = G^{(m)}$ zu setzen und es wird das Element $t_{m-1} \cdots t_0 x$ kanonisiert.

Abbildung 2.1.: Ausschnitt aus dem Backtrackbaum zu Algorithmus 2.6



Der Aufruf für das Transversalenelement t_m ergab bereits

$$g_{opt}x = t_{r-1}\cdots t_0 x \in (\cdots((Can_{id})^{\uparrow G^{(r-1)}})\cdots)^{\uparrow G^{(m+1)}}(H^{(m+1)})$$
$$\Rightarrow g_r x = s_{r-1}\cdots s_m t_{m-1}\cdots t_0 x \in (\cdots((Can_{id})^{\uparrow G^{(r-1)}})\cdots)^{\uparrow G^{(m+1)}}(H^{(m+1)}).$$

Alle weiteren $H^{(m+1)}$–abhängigen Elemente zu $g_r x$ brauchen nicht untersucht zu werden, da bereits $g_r x$ ein $H^{(m+1)}$– kanonisches Element ist, was in unserem Falle minimal in der Bahn heißt. Also gilt für alle Elemente $g \in \left(H^{(m+1)}s_m \cdots s_0\right)$ des darunter liegenden Teilbaums: $gx \geq g_{opt}x$. Nach Algorithmus 2.5 auf Seite 23 für das Heben eines Kanonisierers reicht es bereits, den angegebenen Erzeuger $g_{opt}^{-1}g_r$ zu S hinzuzufügen. Wir können also den Teilbaum $H^{(m+1)}s_m \cdots s_0$ abschneiden, was nichts anderes heißt, als die Betrachtung auf Ebene $m+1$ fortzusetzen. □

Die Abbildung 2.1 stellt nochmals das Vorgehen des Algorithmus 2.6 an einem Knoten $g_i = t_{i-1}\cdots t_0$ dar.

2.3.5 Bemerkung. Die Zeile 14 des Algorithmus 2.6 bedeutet nichts weiteres, als zum letzten, gemeinsamen Vorgänger im Backtrackbaum zurückzuspringen und dessen nächsten, unbetrachteten Sohn – falls vorhanden – bzw. einen unbetrachteten Sohn eines Vorgängers gemäß der Backtrackvorschrift zu untersuchen.

Wir führen die beiden Ideen der Kanonisierung mittels Homomorphieprinzip und des Hebens entlang einer Untergruppenkette $G = G^{(0)} \geq G^{(1)} \geq \ldots G^{(r)} = \{\mathrm{id}\}$ zusammen. Wir werden zwischen jeden Schritt des Hebens eine Folge $(f_{i,j})_{j\in j_i}$ von

$G^{(i)}$–Homomorphismen vorschreiben:

$$f_{i,j} : X \to Y_{i,j}, i \in r, j \in j_i.$$

Die Notation als Folge werden wir analog zu oben auch durch den $G^{(i)}$–Homomorphismus

$$\begin{aligned} \overline{f}_i : X &\to \bigtimes_{j \in j_i} Y_{i,j} \\ x &\mapsto (f_{i,0}(x), \ldots, f_{i,j_i-1}(x)) \end{aligned}$$

ersetzen.

2.3.6 Satz (Zusammenfassung)**.** *Es sei $(X, \leq)$ eine totalgeordnete G–Menge und $G = G^{(0)} \geq \cdots \geq G^{(r)} = \{\mathrm{id}\}$ eine Untergruppenkette. Für jedes $i \in r$ sei außerdem eine Folge $(f_{i,j})_{j \in j_i}$ von $G^{(i)}$–Homomorphismen gegeben:*

$$f_{i,j} : X \to Y_{i,j}, i \in r, j \in j_i.$$

Außerdem seien die Mengen $Y_{i,j}$ totalgeordnet und das System kanonischer Transversalen $Can_{Y_{i,j}}$ gleich Can_{min} auf $Y_{i,j}$ mit einem zugehörigen Kanonisierer $can_{Y_{i,j}}$. Entsprechend wählen wir für $\overline{Y}_i := \bigtimes_{j \in j_i} Y_{i,j}$ die lexikographische Ordnung auf den Vektoren, d.h. wir betrachten auch hier $Can_{\overline{Y}_i} = Can_{min}$ als System kanonischer Transversalen mit dem Kanonisierer gemäß Algorithmus 2.3. Es gilt:
Der Algorithmus 2.7 ist ein Kanonisierer zu dem System kanonischer Transversalen

$$((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j \in j_{r-1}}} \cdots)^{\uparrow G^{(0)}})_{(f_{0,j})_{j \in j_0}}.$$

Das Heben von $G^{(i)}$ zu $G^{(i-1)}$ führen wir mittels der folgenden Totalordnung

$$\begin{aligned} x \preceq x' :\Longleftrightarrow &\Big(\exists i \in r : \overline{f}_i(x) < \overline{f}_i(x') \wedge \forall i' < i : \overline{f}_i(x) = \overline{f}_i(x')\Big) \\ &\vee \Big(\forall i \in r : \overline{f}_i(x) = \overline{f}_i(x') \wedge x \leq x'\Big) \end{aligned}$$

auf X durch, um geeignete Testfunktionen „$\prec_i$“zum Abschneiden von Teilbäumen zur Verfügung zu stellen. Wir setzen dementsprechend für $i \in r$:

$$x \prec_i x' :\Longleftrightarrow \overline{f}_i(x) < \overline{f}_i(x') \text{ und } x \prec_r x' :\Longleftrightarrow x < x'.$$

Die H–kanonischen Repräsentanten sind genau die minimalen Elemente einer jeden Bahn bezüglich „$\preceq$“:

$$x \in ((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j \in j_{r-1}}} \cdots)^{\uparrow G^{(0)}})_{(f_{0,j})_{j \in j_0}}(H) \Longleftrightarrow \forall h \in H : x \preceq hx$$

Beweis. Wir zeigen nur die letzte Behauptung bezüglich der Minimalität der H–kanonischen Repräsentanten durch Induktion nach der Länge der Untergruppenkette.

$\underline{r=1}$: Die Untergruppenkette sei $G = G^{(0)} \geq G^{(1)} = \{\mathrm{id}\}$:

$$\begin{aligned}
& x \in ((Can_{\mathrm{id}})^{\uparrow G^{(r-1)}})_{\overline{f}_0} \\
\Longleftrightarrow & \overline{f}_0(x) \in Can_{\overline{Y}_0}(H) \wedge x \in (Can_{id})^{\uparrow G}(H_{\overline{f}_0(x)}) \\
\Longleftrightarrow & \overline{f}_0(x) \in Can_{\overline{Y}_0}(H) \wedge x \in Can_{\mathrm{id}}(H^{(1)}_{\overline{f}(x)}) \wedge \\
& \forall h \in H_{\overline{f}_0(x)}, hx \in Can_{\mathrm{id}}(H^{(1)}_{\overline{f}(x)}) : x \preceq hx
\end{aligned}$$

Nun ist aber $Can_{\overline{Y}_0} = Can_{min}$, also $\overline{f}(x) \leq h\overline{f}(x) = \overline{f}(hx), \forall h \in H$. Die Bedingung $x' \in Can_{\mathrm{id}}(H^{(1)}_{\overline{f}(x')})$ ist ohnehin für alle $x' \in X$ erfüllt.

$$\begin{aligned}
& x \in ((Can_{\mathrm{id}})^{\uparrow G^{(r-1)}})_{\overline{f}_0} \\
\Longleftrightarrow & \overline{f}_0(x) \in Can_{\overline{Y}_0}(H) \wedge \forall h \in H_{\overline{f}_0(x)} : x \preceq hx \\
\Longleftrightarrow & \forall h \in H : x \preceq hx
\end{aligned}$$

$\underline{r > 1}$: Die Untergruppenkette sei $G = G^{(0)} \geq \cdots \geq G^{(r)} = \{\mathrm{id}\}$. Es gilt für $x \in X$:

$$\begin{aligned}
& x \in ((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}} \cdots)^{\uparrow G^{(0)}})_{(f_{0,j})_{j\in j_0}}(H) \\
\Longleftrightarrow & \overline{f}_0(x) \in Can_{\overline{Y}_0}(H) \\
& \wedge x \in ((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}} \cdots)^{\uparrow G^{(0)}})(H_{\overline{f}_0(x)}) \\
\Longleftrightarrow & \overline{f}_0(x) \in Can_{\overline{Y}_0}(H) \\
& \wedge x \in ((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}} \cdots)^{\uparrow G^{(1)}})_{(f_{1,j})_{j\in j_1}}(H^{(1)}_{\overline{f}_0(x)}) \\
& \wedge \forall h \in H_{\overline{f}_0(x)}, \\
& \quad hx \in ((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}} \cdots)^{\uparrow G^{(1)}})_{(f_{1,j})_{j\in j_1}}(H^{(1)}_{\overline{f}_0(x)}) : \\
& \quad x \preceq hx
\end{aligned}$$

Da nun jeder $H^{(1)}_{\overline{f}_0(x)}$–kanonische Repräsentant nach Induktionsvoraussetzung minimal in seiner Bahn bezüglich der lexikographischen Ordnung auf $\overline{Y}_1 \times \ldots \times \overline{Y}_{r-1} \times X$ ist, gilt, falls wir γ für die $H^{(1)}_{\overline{f}_0(x)}$–kanonisierende Abbildung schreiben

$$\begin{aligned}
& x \in ((\cdots((Can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}} \cdots)^{\uparrow G^{(0)}})_{(f_{0,j})_{j\in j_0}}(H) \\
\Longleftrightarrow & \forall h \in H : \overline{f}_0(x) \leq \overline{f}_0(hx) \wedge \forall h' \in H_{\overline{f}_0(x)} : x \preceq \gamma(h'x)h'x \preceq h'x \\
\Longleftrightarrow & \forall h \in H : x \preceq hx
\end{aligned}$$

□

2.3.7 Definition. Wir definieren zu der Untergruppe $H \leq G, x \in X$ noch induktiv die auftretenden Stabilisatoren während des Backtrackdurchlaufs gemäß Algorithmus 2.7:

- $H_x^{(0,0)} := H$
- $\forall i \in n, j < j_i : H_x^{(i,j+1)} := \left(H_x^{(i,j)}\right)_{f_{i,j}(x)}$
- $\forall i \in n : H_x^{(i+1,0)} := H_x^{(i,j_i)} \cap S_n^{(i+1)}$

2.3.8 Bemerkung. Für ein g_i auf Ebene i wird durch $\{T_{i-1}^{(i)}, \ldots, T_{i-1}^{(r-1)}\}$ die Gruppe $H_{g_i x}^{(i,0)}$ verwaltet. Durch den Aufruf von $can_{\overline{Y}_i}$ wird dann ein $h = h_{j_i-1} \cdots h_0 \in H_{g_i x}^{(i,0)}$ zur Kanonisierung in $\overline{Y}_i$ über die Untergruppenkette

$$H_{g_i x}^{(i,0)} \geq H_{h_0 g_i x}^{(i,1)} \geq \ldots \geq H_{h_{j_i-1} \cdots h_0 g_i x}^{(i,j_i)}$$

berechnet, $h_j \in H_{h_{j-1} \cdots h_0 g_i x}^{(i,j)}$– siehe Algorithmus 2.3. Die Ausgabe $\{T_i^{(i)}, \ldots, T_i^{(r-1)}\}$ des Kanonisierers im Bildbereich $can_{\overline{Y}_i}$ sei in diesem Falle eine Transversale von $H_{hg_i x}^{(i,j_i)}$, also zu dem Stabilisator des ausgegeben, kanonischen Elements, und nicht zu $g_i x$.

Anschließend durchläuft das Backtrackverfahren auf Ebene $i+1$ eine Transversale $T_i^{(i)}$ von $H_{hg_i x}^{(i+1,0)} \backslash H_{hg_i x}^{(i,j_i)}$. Da für ein $t \in H_{hg_i x}^{(i,j_i)}$ die Gruppen $H_{hg_i x}^{(i,j_i)}$ und $H_{thg_i x}^{(i,j_i)}$ gleich sind, gilt insbesondere

$$H_{hg_i x}^{(i+1,0)} = H_{thg_i x}^{(i+1,0)}.$$

2.3.9 Bemerkung. Die Abarbeitung der Testfunktionen $\prec_i$ sollte parallel zur Kanonisierung mittels $can_{\overline{Y}_i}$ stattfinden. Stellt sich bei der Kanonisierung mittels Homomorphieprinzip für ein $j \in j_i$ und h ein kanonisierendes Element erstmalig $f_{i,j}(hg_i x) > f_{i,j}(g_{opt} x)$ heraus, so kann abgebrochen werden. Ist $f_{i,j}(hg_i x) < f_{i,j}(g_{opt} x)$, so wird $g_{opt} \leftarrow g_i$ gesetzt.

Algorithmus 2.7 $((\cdots((can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}}\cdots)^{\uparrow G^{(0)}})_{(f_{0,j})_{j\in j_0}}$– Version 1

Input: $x \in X$ eine diskrete Struktur
Input: $\{T^{(0)}, \ldots, T^{(r-1)}\}$ Rechtstransversalen der Untergruppenkette von H
Output: $g \in H$ kanonisierendes Element
Output: $\langle S \rangle = H_x$

1: $g_{opt} \leftarrow \mathrm{id}$;
2: $S \leftarrow \emptyset$;
3: $\big(h, \{T_0^{(0)}, \ldots, T_0^{(r-1)}\}\big) \leftarrow can_{\overline{Y}_0}\big(\{T^{(0)}, \ldots, T^{(r-1)}\}, \overline{f}_0(x)\big)$;
4: $g_0 \leftarrow h$;
5: $\overline{T}^{(0)} \leftarrow T_0^{(0)}$;
6: **while** Durchlaufe Backtrackbaum ohne Wurzel zu $\{\overline{T}^{(0)}, \ldots, \overline{T}^{(r-1)}\}$ mittels Rechtstransversalenergänzung **do**
7: // Die Transversalen $\overline{T}^{(i)}$ werden während des Durchlaufs passend gesetzt
8: $i \leftarrow$ Ebene des aktuellen Knotens;
9: $j \leftarrow$ Index der eingehenden Kante;
10: $g_i \leftarrow \overline{T}^{(i-1)}[j] \cdot g_{i-1}$;
11: $\big(h, \{T_i^{(i)}, \ldots, T_i^{(r-1)}\}\big) \leftarrow can_{\overline{Y}_i}\big(\{T_{i-1}^{(i)}, \ldots, T_{i-1}^{(r-1)}\}, \overline{f}_i(g_i x)\big)$;
12: $g_i \leftarrow h g_i$;
13: $\overline{T}^{(i)} \leftarrow T_i^{(i)}$;
14: **if** $g_i x \succ_i g_{opt} x$ **then**
15: **continue on level** i; // Überspringe darunterliegenden Teilbaum
16: **end if**
17: **if** $i = r \wedge g_i x = g_{opt} x \wedge g_i x \neq g_{opt} x$ **then**
18: $S \leftarrow S \cup \{g_{opt}^{-1} g_i\}$;
19: $m \leftarrow \max\{m' \leq r \mid g_i g_{opt}^{-1} \in G^{(m')}\}$;
20: **continue on level** $m + 1$;
21: **end if**
22: **if** $g_i x \prec_i g_{opt} x$ **then**
23: $g_{opt} \leftarrow g_i$;
24: **end if**
25: **if** $i < r$ **then**
26: **continue on level** $i + 1$;
27: **else**
28: **continue on level** i;
29: **end if**
30: **end while**
31: **return** (g_{opt}, S);

2.4. Abschneiden von Teilbäume mit vollständigen Labelled Branchings

Wir setzen in diesem Abschnitt voraus, dass $G \leq S_n$ gelte. Den Backtrackbaum durchlaufen wir weiterhin wie in Algorithmus 2.7. Wir wollen uns bei der Untersuchung aber nur noch auf Elemente einer Transversalen T von $G/\langle S \rangle$ beschränken, da die Mengen

$$G(x) \quad \text{und} \quad \{tx \mid t \in T\}$$

offensichtlich gleich sind. Wir benötigen also einen effektiven Test, ob in der Rechtsnebenklasse $G^{(i)}g_i$ (also im Teilbaum unterhalb von g_i auf Ebene i) Transversalenelemente von $G/\langle S \rangle$ liegen. Ist dies nicht der Fall, so können wir den gesamten Teilbaum abschneiden, da die Bahnenelemente $G^{(i)}(g_i x)$ über andere Permutationen erreicht werden.

Als Untergruppenkette betrachten wir nun die Stabilisatorkette

$$G = G^{(0)} \geq G^{(1)} \geq \ldots \geq G^{(n-1)} \geq G^{(n)} = \{\mathrm{id}\}$$

mit $G^{(i)} := G_{(b_0,\ldots,b_{i-1})}, i \in n$, welche durch die Wahl einer Basis $\overrightarrow{b} := (b_0, \ldots, b_{n-1})$ festgelegt wurde.

Außerdem erhalten wir durch die Festlegung der Basis eine Ordnung auf n mittels

$$b_i \leq b_j :\Longleftrightarrow i \leq j.$$

2.4.1 Bemerkung. Damit sind auch die Permutationen durch die lexikographische Anordnung der Listenschreibweise geordnet. Es gelte für $g, h \in S_n$:

$$g \leq h :\Longleftrightarrow \big(g(b_0), \ldots, g(b_{n-1})\big) \leq \big(h(b_0), \ldots, h(b_{n-1})\big).$$

2.4.2 Definition. Wir definieren den Typ eines $g \in G$ als das Paar von Indizes $\mathrm{typ}(g) := (i, j)$ mit:

$$i = max\{i' \in n \mid g \in G^{(i')}\} \quad \text{und} \quad g(b_i) = b_j.$$

Der Index i heißt Stablisatorindex und j der Transversalenindex von g.

2.4.3 Hilfssatz. *Sei $g \in G$ und $\mathrm{typ}(g) = (i, j)$. Ist $g \neq id$, so gilt $i < j$, und für die Identität gilt:* $\mathrm{typ}(\mathrm{id}) = (n-1, n-1)$. *Definieren wir für zwei Typen $(i, j), (i', j')$ die Ordnungsrelation*

$$(i, j) \leq (i', j') :\Longleftrightarrow i > i' \vee (i = i' \wedge j \leq j'),$$

so gilt:

$$g \leq g' \Longrightarrow \mathrm{typ}(g) \leq \mathrm{typ}(g')$$

2.4.4 Definition (Labelled Branching)**.** Wir nennen ein Tripel $(\overrightarrow{b}, \Gamma, \gamma)$ ein Labelled Branching von G, falls die folgenden Bedingungen erfüllt sind

1. $\overrightarrow{b} = (b_0, \ldots, b_{n-1})$ ist eine Basis von G auf n,
2. Γ ist ein Branching mit Knotenmenge n, dh. ein azyklischer Digraph auf n mit maximalen Eingangseckengrad 1. Weiterhin gelte für jede Kante (b_i, b_j) die Bedingung $i < j$,
3. $\gamma = (\gamma_0, \ldots, \gamma_{n-1})$ sei ein Vektor von Permutationen $\gamma_i \in S_n, i \in n$ mit:
   - Ist (b_i, b_j) eine Kante in Γ, so ist $\sigma_{ij} := \gamma_j^{-1}\gamma_i$ vom Typ $\mathrm{typ}(\sigma_{ij}) = (i, j)$,
   - Die Menge $E := \{\sigma_{ij} \mid (b_i, b_j) \text{ ist Kante in } \Gamma\}$ ist ein Erzeugendensystem von G.

Wir wollen nun kurz beschreiben wie man von einem beliebigen Erzeugendensystem F zu dem Labelled Branching $(\overrightarrow{b}, \Gamma, \gamma)$ kommt. Ausführlichere Beschreibungen finden sich in [Gug05] oder [Jer86].

Wir definieren zu F den gerichteten Graphen $\Gamma(n, \overrightarrow{b}, F)$ mit Knotenmenge n und Kantenmenge $V := \{(b_i, b_j) \mid \exists g \in F : (i, j) = \mathrm{typ}(g)\}$. Weiterhin beschriften wir die Kante mit den Gruppenelementen $g \in F$. Entfernt man gegebenenfalls den trivialen Erzeuger id aus F, so liegen nur noch Kanten vor mit $b_i < b_j$, insbesondere ist der Graph also ohne Kreise. Wir wenden nun iterativ und solange wie möglich eine der folgenden beiden Ersetzungen für F an:

1. Sind $g, h \in F$ mit $g \neq h$ und $\mathrm{typ}(g) = \mathrm{typ}(h)$, so ersetzen wir F durch das äquivalente Erzeugendensystem $F \setminus \{g\} \cup \{h^{-1}g\}$. Der Typ von $h^{-1}g$ ist dann echt kleiner als der von g.
2. Sind $g, h \in F$ und $\mathrm{typ}(g) = (i_0, j), \mathrm{typ}(h) = (i_1, j)$ mit $i_0 < i_1$, so können wir $g' := h^{-1}g$ bilden. Der Typ von g' ist (i_0, i_1) mit $i_1 < j$. Wir ersetzen in F wieder g durch das Element g' von echt kleinerem Typ.

Da durch diese beiden Ersetzungsschritte stets das Erzeugnis konstant bleibt, jedoch stets ein Erzeuger durch einen Erzeuger von echt kleinerem Typ ersetzt wird, muss das Verfahren terminieren. Nach Abschluss des Verfahrens mündet in jedem Knoten also nur eine Kante ein. $(\overrightarrow{b}, \Gamma, \gamma)$ ist also ein Branching. Wir definieren für $g \in F$:

$$\sigma_{ij} := g \Longleftrightarrow \mathrm{typ}(g) = (i, j)$$

Sei $G' > G$ eine Obermenge von G und T eine Transversale der Operation von G' auf n. Wir ordnen jeder Wurzel b_i (Knoten mit Eingangseckengrad 0) in $\Gamma(n, \overrightarrow{b}, F)$ ein kanonisierendes Element γ_i zu, also $\gamma \in G' : \gamma_i x_i \in T$. Für die restlichen Knoten setzt man induktiv: $\gamma_j := \gamma_i \sigma_{ij}^{-1}$. Dann ist $\gamma : x_i \mapsto \gamma_i$ eine kanonisierende Abbildung von G' auf n, und γ erfüllt die Eigenschaften des Punktes 3. in der Definition des Labelled Branchings.

2.4.5 Definition. Ist Γ ein Branching auf n, so definiert es eine Teilordnung auf den Elementen. Wir schreiben für zwei Elemente $b_i, b_j \in n$:

$$b_i \preceq b_j :\Longleftrightarrow \text{Es gibt in } \Gamma \text{ einen (gerichteten) Pfad von } b_i \text{ nach } b_j$$
$$b_i \dot{\preceq} b_j :\Longleftrightarrow \text{Es gibt in } \Gamma \text{ eine (gerichtete) Kante von } b_i \text{ nach } b_j$$

Ist $b_i \dot{\preceq} b_j$, so definieren wir $\text{father}_j := i$. Falls b_j eine Wurzel des Branchings ist, so setzen wir $\text{father}_j := -1$. Im Folgenden sei angenommen, dass das Branching über einen solchen Vektor $\text{father} = (\text{father}_0, \ldots, \text{father}_{n-1})$ implementiert sei.

2.4.6 Definition. Ist für jedes $i \in n$ die Menge $T^{(i)} := \{\sigma_{ij} \mid x_i \preceq x_j\}$ eine Linkstransversale von $G^{(i)}/G^{(i+1)}$, so heißt das Labelled Branching vollständig.

Wie man vollständige Labelled Branchings effektiv aufbaut, wird in der oben genannten Literatur beschrieben. Da für das hier entwickelten Programm auf bereits vorhandene Implementierungen der Algorithmen für Labelled Branchings zurückgegriffen werden konnte, nehmen wir auch an dieser Stelle an, dass wir einen Algorithmus $sift()$ (nach [Jer86]) zur Verfügung haben, um ein vorhandenes Erzeugendensystem um einen Erzeuger zu erweitern und ein (möglicherweise nicht vollständiges) Labelled Branching zu konstruieren. Außerdem sei ein Algorithmus *complete*() gegeben, der ein Labelled Branching zu einem vollständigen Labelled Branching umformt.

Wir werden nun die Menge $\langle S \rangle$ der bisher bekannten Automorphismen eines $x \in X$ über ein vollständiges Labelled Branching verwalten und stets aktualisieren.

2.4.7 Satz. *Sei $G \leq S_n$ und $(\overrightarrow{b}, \Gamma, \gamma)$ ein vollständiges Labelled Branching zu G. Ein $\pi \in S_n$ ist genau dann lexikographisch minimal in der Linksnebenklasse πG, falls π eine topologische Anordnung von Γ ist, dh. für $b_i, b_j \in n$ gilt:*

$$b_i \preceq b_j \Longrightarrow \pi(b_i) \leq \pi(b_j)$$

Beweis. Wir zeigen die Kontraposition:
„$\Rightarrow$“: Angenommen π sei keine topologische Anordnung von Γ, dann existiert eine Kante (b_i, b_j) in Γ mit $\pi(b_i) > \pi(b_j)$. Es sei $\sigma_{ij} = \gamma_j^{-1}\gamma_i$ die Beschriftung dieser Kante. Die Permutation $\pi\sigma_{ij} \in \pi G$ ist lexikographisch kleiner als π, da

$$\pi\sigma_{ij}(b_k) = \pi(b_k), \forall 0 \leq k < i$$
$$\pi\sigma_{ij}(b_i) = \pi(b_j) < \pi(b_i)$$

„$\Leftarrow$“: Sei π nicht minimal in der Linksnebenklasse πG. Wir wählen ein $g \in G$ mit $\pi g < \pi$. Der Typ von $g \neq \text{id}$ sei (i, j) mit $i < j$. Somit gilt wieder für alle $0 \leq k < i : \pi g(b_k) = \pi(b_k)$ und $\pi g(b_i) = \pi(b_j) \neq \pi(b_i)$. Da aber πg lexikographisch kleiner als π vorausgesetzt war, ist $\pi(b_i) > \pi(b_j)$.
Es ist aber Γ ein vollständiges Labelled Branching zu G und da $g \in G^{(i)}$ existiert nach Folgerung 1.1.9 auch ein Transversalenelement $t \in T^{(i)}$ mit $t(b_i) = b_j$. Somit muss es einen Pfad von b_i nach b_j geben. Aus diesem Grunde kann π keine topologische Anordnung zu Γ sein. □

2.4.8 Folgerung. *Ist $G \leq G' \leq S_n$ und $(\overrightarrow{b}, \Gamma, \gamma)$ ein vollständiges Labelled Branching zu G, so erhält man durch die folgenden Definition eine Transversale T_Γ der lexikographisch minimalen Elemente der Linksnebenklassen G'/G:*

$$T_\Gamma := \{\pi \in G' \mid \forall b_i, b_j \in n : b_i \preceq b_j \Rightarrow \pi(b_i) \leq \pi(b_j)\}.$$

Wir gehen nun wieder zurück zu unserer Ausgangssituation. Wir wollen die Bahn $G(x)$ eines $x \in X$ mit dem Backtrackverfahren durch Ergänzung mit Rechtstransversalenelemente der Stabilisatorkette durchlaufen. Die Gruppe $\langle S \rangle \leq Aut(x)$ der bisher bekannten Automorphismen werde über ein vollständiges Labelled Branching $(\overrightarrow{b}, \Gamma, \gamma)$ mit father–Vektor verwaltet. Wir wollen für ein $g_i \in G$ auf Ebene i ausschließen, dass im darunterliegenden Teilbaum $G^{(i)}g_i$ Transversalenelemente aus T_Γ liegen. Der nächste Hilfssatz gibt uns dazu den Test.

2.4.9 Hilfssatz (siehe auch Hilfssatz 3.3.3 [Gug05])**.** *Sei $\pi \in S_n$ und $S_n^{(i)}$ der punktweise Stabilisator von $b_0, \ldots, b_{i-1}$. Dann gibt es in der Rechtsnebenklasse $S_n^{(i)}\pi$ genau dann topologische Anordnungen zu dem vollständigen Labelled Branching $(\overrightarrow{b}, \Gamma, \gamma)$ von G, wenn für alle $j < i$ und $b_k := \pi^{-1}(b_j)$ gilt:*

$$\text{father}_k \geq 0 \Rightarrow \pi(b_{\text{father}_k}) < b_j$$

Beweis. „⇒“: In der Nebenklasse $S_n^{(i)}\pi$ bleiben die Urbilder $\sigma^{-1}(b_l)$ für $l < i$, $\sigma \in S_n^{(i)}\pi$ konstant: Sei $l < i$ beliebig und $b_k = \pi^{-1}(b_l)$. Dann ist für alle $\sigma \in S_n^{(i)}\pi : \sigma(b_k) = b_l$. Ist $k' = \text{father}_k \neq -1$, so ist entweder $\sigma(b_{k'}) < b_l$ konstant für alle $\sigma \in S_n^{(i)}\pi$, oder für alle $\sigma \in S_n^{(i)}\pi$ gilt: $\sigma(b_{k'}) > b_l$. Existiert also eine topologische Anordnung $\sigma \in S_n^{(i)}\pi$, so muss die dafür notwendige Bedingung $\sigma(b_{k'}) < \sigma(b_k) = b_l$ bereits für π erfüllt sein.
„⇐“: Aufgrund der Voraussetzungen bildet die Urbildmenge $I := \pi^{-1}(\{b_0, \ldots, b_{i-1}\})$ ein Ordnungsideal bzgl. der Teilordnung $\preceq$, dh. ist $b_k \in I, b_{k'} \preceq b_k \Rightarrow b_{k'} \in I$. Das Komplement $J := X \setminus I = \{b_l \mid \pi(b_l) \geq b_i\}$ bildet entsprechend einen unteren Teilwald von Γ, beziehungsweise wieder ein Labelled Branching. Wir können J topologisch anordnen bzgl. dem Teilgraphen $\Gamma{\downarrow_J}$ und erhalten so eine topologische Anordnung auf ganz Γ. Die Umsortierung von J entspricht einer Linksmultiplikation mit einem $\sigma \in S_n^{(i)}$, d.h. die gefundene topologische Anordnung liegt in $S_n^{(i)}\pi$. □

2.4.10 Folgerung. *Sei $\pi \in S_n^{(i)}$ und $(\overrightarrow{b}, \Gamma, \gamma)$ ein vollständiges Labelled Branching zu $\langle S \rangle$. Es gilt also:*

$$\big(\exists l < i, b_k := \pi^{-1}(b_l) : father_k \geq 0 \wedge \pi(b_{father_k}) \geq b_l\big) \Rightarrow T_\Gamma \cap S_n^{(i)}\pi = \emptyset.$$

Wir fügen diesen Test in Algorithmus 2.7 ein, brauchen die Bedingung aber wegen der Durchlaufreihenfolge nur für $l = i - 1$ zu testen. Um die Richtigkeit garantieren zu können, müssen wir das Zurückspringen zum letzten gemeinsamen Vorgänger bei Auffinden eines Automorphismus wegfallen lassen. Würden wir dies nicht tun, könnte die folgende Situation eintreten:

- Wir überspringen die Untersuchung eines $g \in G$ nach dem Kriterium aus Hilfssatz 2.4.9, da wir nur das Transversalenelement $h \in g\langle S\rangle$ untersuchen müssen.
- Wir erreichen ein Gruppenelement g' mit $g'x = g_{opt}x$ und brechen die Untersuchung des Teilbaums $G^{(m+1)}g'$ ab.
- Gilt nun $h \in G^{(m+1)}g'$ und h wird in der Backtrackdurchlaufreihenfolge nach g' betrachtet, so wird das Bahnenelement hx möglicherweise nicht erreicht und bleibt von der Untersuchung ausgeschlossen.

2.4.11 Bemerkung. Die Kombination beider Ideen ist nur möglich, falls wir den Baum lexikographisch aufsteigend durchlaufen. In diesem Fall werden dann die Transversalenelemente von $G/\langle S\rangle$ stets zuerst erreicht.

2.4.12 Bemerkung. Durch den Test mittels vollständigen Labelled Branching erhalten wir, sobald alle Automorphismen erreicht wurden ($\langle S\rangle = Aut(x)$) eine Aufwandsreduktion um den Faktor $\big|Aut(x)\big|$. Das Zurückspringen bei Auffinden eines Automorphismus aus Algorithmus 2.7 stellt jedoch keinen Lerneffekt zur Verfügung, es findet zum Beispiel Automorphismen mehrfach. In ungünstigen Fällen sind weiterhin alle Knoten zu betrachten. Desweiteren erhalten wir nicht ein Erzeugendensystem sondern unter Umständen die volle Automorphismengruppe als Rückgabe. Allein diese Tatsache ist aus Speichergründen von großem Nachteil. Aus diesen Gründen bevorzugen wir den Test mittels Hilfssatz 2.4.9 und nehmen den Verlust des Verfahrens nach Algorithmus 2.7 in Kauf.

Wir ersetzen desweiteren noch die Testfunktionen „$\prec_i$“jeder Ebene $i \in \{1, \dots, r\}$ durch ihre ursprüngliche Definitionen und sortieren die Abarbeitung der Fälle nach Blattknoten oder Nichtblattknoten. Außerdem sei die Basis ohne Beschränkung der Allgemeinheit $\overrightarrow{b} = (0, 1, \dots, n-1)$. Wir erhalten schließlich den Algorithmus 2.8.

Algorithmus 2.8 $((\cdots((can_{id})^{\uparrow G^{(r-1)}})_{(f_{r-1,j})_{j\in j_{r-1}}}\cdots)^{\uparrow G^{(0)}})_{(f_{0,j})_{j\in j_0}}$– mit vollst. Labelled Branching

---

Input: $x \in X$ eine diskrete Struktur
Input: $\{T^{(0)}, \dots, T^{(r-1)}\}$ Rechtstransversalen der Untergruppenkette von H
Input: S Gruppe von Automorphismen als vollständiges Labelled Branching
Output: $g \in H$ kanonisierendes Element
Output: $\langle S\rangle = H_x$

1: $\big(h, \{T_0^{(0)}, \ldots, T_0^{(r-1)}\}\big) \leftarrow can_{\overline{Y}_0}\big(\{T^{(0)}, \ldots, T^{(r-1)}\}, \overline{f}_0(x)\big)$;
2: $g_0 \leftarrow h$;
3: $\overline{T}^{(0)} \leftarrow T_0^{(0)}$;
4: **while** Durchlaufe Backtrackbaum ohne Wurzel zu $\{\overline{T}^{(0)}, \ldots, \overline{T}^{(r-1)}\}$ mittels Rechtstransversalenergänzung **do**
5: // Die Transversalen $\overline{T}^{(i)}$ werden während des Durchlaufs passend gesetzt
6: $i \leftarrow$ Ebene des aktuellen Knotens;
7: $j \leftarrow$ Index der eingehenden Kante;
8: $g_i \leftarrow \overline{T}^{(i-1)}[j] \cdot g_{i-1}$;
9: $k \leftarrow g_i^{-1}(i-1)$;
10: **if** $S.father_k \geq 0 \wedge g_i(S.father_k) \geq i-1$ **then**
11: // In diesem Teilbaum gibt es kein Transversalenelement von $H/\langle S \rangle$ nach Hilfssatz 2.4.9
12: **continue on level** i;
13: **end if**
14: $\big(h, \{T_i^{(i)}, \ldots, T_i^{(r-1)}\}\big) \leftarrow can_{\overline{Y}_i}\big(\{T_{i-1}^{(i)}, \ldots, T_{i-1}^{(r-1)}\}, \overline{f}_i(g_i x)\big)$;
15: $g_i \leftarrow h g_i$;
16: $\overline{T}^{(i)} \leftarrow T_i^{(i)}$;
17: **if** $i < r$ **then**
18: **if** $\overline{f}_i(g_i x) > \overline{f}_i(g_{opt} x)$ **then**
19: **continue on level** i; // Überspringe darunterliegenden Teilbaum
20: **end if**
21: **if** $\overline{f}_i(g_i x) < \overline{f}_i(g_{opt} x)$ **then**
22: $g_{opt} \leftarrow g_i$; // Ein neuer Kandidat für das kanonisierende Element
23: **end if**
24: **continue on level** $i+1$;
25: **end if**
26: **if** $i = r$ **then**
27: **if** $g_i x = g_{opt} x$ **then**
28: $S.sift(g_{opt}^{-1} g_i)$; // Ein neuer Automorphismus
29: $S.complete()$;
30: **end if**
31: **if** $g_i x < g_{opt} x$ **then**
32: $g_{opt} \leftarrow g_i$; // Ein neuer Kandidat für das kanonisierende Element
33: **end if**
34: **continue on level** i;
35: **end if**
36: **end while**
37: **return** (g_{opt}, S);

2.5. Vom Kanonisierer zu weiteren Algorithmen

Mit leichten Modifikationen des Algorithmus 2.8 können wir zwei verwandte Fragestellungen ebenfalls beantworten. Diese können zwar ebenfalls mit Hilfe des Kanonisierers bearbeitet werden, jedoch möglicherweise mit viel zu großem Aufwand.

2.5.1. Bestimmung der Automorphismengruppe

Wir erhalten als Nebenprodukt der Kanonisierung mit Algorithmus 2.8 ein Erzeugendensystem der Automorphismengruppe H_x. Wollen wir jedoch nur die Automorphismengruppe bestimmen, so nutzen wir die Informationen auf den einzelnen Ebenen nur unzureichend. Aus dem Homomorphieprinzip erhalten wir für den G–Homomorphismus $f : X \to Y$: $H_x \subseteq H_{f(x)}$. Wir können in diesem Fall Teilbäume nicht nur Abschneiden, falls die Bedingung

$$\overline{f}_i(g_{opt}x) < \overline{f}_i(g_i x)$$

erfüllt ist, sondern bereits bei Ungleichheit

$$\overline{f}_i(g_{opt}x) \neq \overline{f}_i(g_i x).$$

Entsprechend ersetzen wir die Zeile 18 durch die Ungleichheit und entfernen die Zeilen 21 bis 23 sowie die Zeilen 31 bis 33 in Algorithmus 2.8. Das Element g_{opt} initialisieren wir, sobald wir einen Blattknoten erreichen, mit der vorliegenden Permutation. Wir erhalten somit einen Algorithmus zur Berechnung der Automorphismengruppe.

2.5.1 Bemerkung. Nun wäre zu erwarten, dass dieser Algorithmus im Laufzeitverhalten günstiger ist, da ein weiteres Kriterium zum Abschneiden von Teilbäumen zur Verfügung steht. Bei den Laufzeitanalysen für den hier entwickelten Kanonisierer – siehe Anhang A.1 – ergab sich jedoch ein überraschendes Ergebnis: Der auf die Bestimmung der Automorphismengruppe zugeschnittene Algorithmus arbeitet im Durchschnitt langsamer als der Kanonisierer. Für eine Begründung sei hierbei ebenfalls auf den Anhang A.3 verwiesen.

2.5.2. Test auf Äquivalenz

Den Test auf H–Abhängigkeit von zwei Elementen $x_0, x_1 \in X$ können wir ebenfalls bereits mit Hilfe des Kanonisierers durchführen. Wir müssen hierzu nur für die berechneten kanonischen Repräsentanten x_0^{opt}, x_1^{opt} die Gleichheit testen. Auch hier scheint ein Aufruf des Kanonisierers für das Element x_1 völlig unnötig und für das Element x_0 nur bedingt sinnvoll. Wir müssen in Algorithmus 2.8 vielmehr die Vergleiche mit dem optimalen Kandidaten $g_{opt}x_1$ durch Vergleiche mit einem Blattknoten gx_0 aus dem Backtrackbaum zu x_0 ersetzen. Wir können nicht direkt mit x_0 vergleichen, da dessen Bilder bezüglich der Homomorphismen nicht minimiert sind.

Während des Algorithmus können wir folgende Fälle unterscheiden:

- Sobald wir einen Knoten g_i erreichen mit $g_i x_1 = g x_0$, können wir die Äquivalenz der beiden Elemente bestätigen.
- Liegt der Spezialfall $g x_0 = x_0^{opt}$ vor und gilt: $\overline{f}_i(g_i x_1) < \overline{f}_i(x_0^{opt})$, so ist

  $$g_i x_1 \prec x_0^{opt} \preceq h x_0, \forall h \in H \Rightarrow \forall h \in H : g_i x_1 \neq h x_0$$

  und damit

  $$H(x_0) \neq H(x_1).$$

  Wir können den Test bereits an dieser Stelle abbrechen und als Antwort die Nichtäquivalenz der beiden Elemente zurückgeben.
- Gilt $\overline{f}_i(g_i x_1) \neq \overline{f}_i(g x_0)$, brauchen wir keine weitere Untersuchungen für den Teilbaum $H^{(i)}(g_i x_1)$ durchzuführen, da

  $$\forall h \in H^{(i)} : h g_i x_1 \neq g x_0$$

  vorliegt. In diesem Teilbaum finden wir also keinen Zeugen für die Äquivalenz.

Haben wir den Backtrackbaum ohne Äquivalenzaussage mit diesen Kriterien vollständig durchlaufen, so gilt

$$\forall h \in H : h x_1 \neq g x_0 \Rightarrow H(x_1) \neq H(x_0).$$

Wir können auch hier den Test mit einer negativen Antwort beenden. Wir fassen das Vorgehen in Algorithmus 2.9 zusammen.

2.5.2 Bemerkung. Starten wir den Test mit einem beliebigen Blattknoten $g x_0$, anstatt zuerst x_0^{opt} zu berechnen und mit diesen die Vergleiche durchzuführen, so erhalten wir analog zu oben für den implementierten Kanonisierer eine erheblich längere, durchschnittliche Laufzeit. Die Begründung ist die gleiche und findet sich in Abschnitt A.3 des Anhangs.

Algorithmus 2.9 *equivalent*(H, x_0, x_1, g) – Äqivalenztest

Input: $x_0, x_1 \in X$ zwei diskrete Strukturen
Input: $g \in H$ sd. gx_0 ein Blattknoten des Backtrackbaums zu x_0 und H
Input: $\{T^{(0)}, \ldots, T^{(r-1)}\}$ Rechtstransversalen der Untergruppenkette von H
Output: $(is\_equivalent, \gamma) \in \mathbb{B} \times H$ mit

- $is\_equivalent = false$, falls $H(x_0) \neq H(x_1)$ (γ ist dann ohne Bedeutung),
- $is\_equivalent = true$, falls $H(x_0) = H(x_1)$, dann ist $\gamma \in H$ sd. $x_0 = \gamma x_1$.

1: $\big(h, \{T_0^{(0)}, \ldots, T_0^{(r-1)}\}\big) \leftarrow can_{\overline{Y}_0}\big(\{T^{(0)}, \ldots, T^{(r-1)}\}, \overline{f}_0(x_1)\big)$;
2: $g_0 \leftarrow h$;
3: $\overline{T}^{(0)} \leftarrow T_0^{(0)}$;
4: **while** Durchlaufe Backtrackbaum ohne Wurzel zu $\{\overline{T}^{(0)}, \ldots, \overline{T}^{(r-1)}\}$ mittels Rechtstransversalenergänzung **do**
5: // Die Transversalen $\overline{T}^{(i)}$ werden während des Durchlaufs passend gesetzt
6: $i \leftarrow$ Ebene des aktuellen Knotens;
7: $j \leftarrow$ Index der eingehenden Kante;
8: $g_i \leftarrow \overline{T}^{(i-1)}[j] \cdot g_{i-1}$;
9: $\big(h, \{T_i^{(i)}, \ldots, T_i^{(r-1)}\}\big) \leftarrow can_{\overline{Y}_i}\big(\{T_{i-1}^{(i)}, \ldots, T_{i-1}^{(r-1)}\}, \overline{f}_i(g_i x_1)\big)$;
10: $g_i \leftarrow h g_i$;
11: $\overline{T}^{(i)} \leftarrow T_i^{(i)}$;
12: **if** $i < r$ **then**
13: **if** $\overline{f}_i(g_i x_1) \neq \overline{f}_i(g x_0)$ **then**
14: **continue on level** i; // Überspringe darunterliegenden Teilbaum
15: **end if**
16: **continue on level** $i + 1$;
17: **end if**
18: **if** $i = r$ **then**
19: **if** $g_i x_1 = g x_0$ **then**
20: **return** (**true**, $g^{-1} g_i$); // Die Elemente sind H–abhängig
21: **end if**
22: **continue on level** i;
23: **end if**
24: **end while**
25: **return** (**false**, NULL); // Baum ohne Aussage vollständig durchlaufen

3. Die Isometrieklassen als Bahnen der symmetrischen Gruppe

Ziel dieses Kapitels ist es, die semilinearen Isometrieklassen von (n,k) – Codes mittels der Bahnenmenge der symmetrischen Gruppe S_n auf einer geeigneten Menge zu beschreiben. Damit ermöglichen wir die Anwendung des in Kapitel 2 gewonnenen Algorithmus 2.8 auch für größere Parameter (n,k), da die Gruppenordnung und somit die Größe des Backtrackbaums stark reduziert wurde. Der verbliebene Anteil der Gruppenoperation geht in die Menge ein, auf der operiert wird. Genauer gesagt schreiben wir sie wiederum als Bahnenmenge einer weiteren Gruppenoperation, jedoch ist diese ohne aufwendigen Kanonisierungsalgorithmus zu verwalten, siehe hierzu Kapitel 4.

3.1. Lineare Isometrieklassen linearer Codes

Sei C ein (n,k,d,q) – Code und Γ eine beliebige Generatormatrix von C. Betrachten wir die Menge aller Generatormatrizen der zu C linear isometrischen Codes. Wie bereits weiter oben gesehen, können wir alle Generatormatrizen eines Codes durch Rechtsmultiplikation mit invertierbaren $k \times k$–Matrizen an eine beliebige Generatormatrix des Codes erreichen. Umgekehrt erhalten wir alle zu C linear isometrischen Codes durch Rechtsmultiplikaton von Γ mit monomialen Matrizen $M_{(\varphi;\pi)} \in M_n(q)$.
Es operieren also sowohl $GL_k(q)$ als auch $M_n(q)$ auf $GF(q)^{k\times n}$ und die Operationen sind wegen der Kommutativität der Matrizenmultiplikation vertauschbar:

$$A \cdot \left(\Gamma \cdot M^T_{(\varphi;\pi)}\right) = A \cdot \Gamma \cdot M^T_{(\varphi;\pi)} = \left(A \cdot \Gamma\right) \cdot M^T_{(\varphi;\pi)}.$$

Somit ist die Menge der Generatormatrizen aller zu C linear isometrischen Codes gleich der Bahn

$$(GL_k(q) \times M_n(q))(\Gamma)$$

der Operation von $GL_k(q) \times M_n(q)$ auf $GF(q)^{k\times n}$. Nun können wir statt der monomialen Matrizen die dazu isomorphe Gruppe $GF(q)^* \wr_n S_n$ auf den $k \times n$–Matrizen operieren lassen.

Ein Kranzprodukt $H \wr_X G$ ist ein spezielles semidirektes Produkt $H^X \rtimes_\theta G$ mit $\theta(g)(\varphi) = \varphi_g$. Wir ersetzen $GF(q)^* \wr_n S_n$ durch die äquivalente Konstruktion mittels semidirekten Produkt $GF(q)^{*n} \rtimes_{\theta_1} S_n$. Wir erhalten für die ursprüngliche Gruppe:

$$GL_k(q) \times M_n(q) \simeq GL_k(q) \times \left(GF(q)^{*n} \rtimes_{\theta_1} S_n\right).$$

Definieren wir $\theta := \big(\theta_0, \theta_1\big)$ mit $\theta_0 : S_n \to Aut(GL_k(q)), \pi \mapsto id_{GL_k(q)}$ so können wir neu klammern und erhalten

$$GL_k(q) \times \big(GF(q)^{*n} \rtimes_{\theta_1} S_n\big) = \big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta} S_n.$$

3.1.1 Satz. *Die Gruppe* $\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta} S_n$ *mit*

$$\theta : S_n \to Aut(GL_k(q) \times GF(q)^{*n}), \pi \mapsto \big(id_{GL_k(q)}, \theta_1(\pi)\big)$$

operiert auf der Menge der $k \times n$*– Matrizen durch:*

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta} S_n\Big) \times GF(q)^{k\times n} \to GF(q)^{k\times n},$$
$$\big((A, \varphi; \pi), \Gamma\big) \mapsto A \cdot \Gamma \cdot M^T_{(\varphi;\pi)}.$$

Die Bahn einer Generatormatrix Γ *eines linearen* (n, k) *– Codes* C *unter dieser Gruppenoperation entspricht der Menge aller Generatormatrizen der zu* C *linear isometrischen Codes.*

Beweis. Siehe vorherige Diskussion. □

Da die Multiplikation der $k \times n$–Matrizen mit invertierbaren Matrizen den Rang der Matrix erhält, können wir die Operation auf die Matrizen $GF(q)^{k\times n,k}$ mit vollem Zeilenrang k einschränken.

3.1.2 Satz. *Die Menge der linearen Isometrieklassen von* (n, k) *– Codes ist bijektiv zu der Bahnenmenge*

$$S_n \backslash\backslash \Big(\big(GL_k(q) \times GF(q)^{*n}\big) \backslash\backslash GF(q)^{k\times n,k}\Big).$$

Beweis. Nach obiger Diskussion ist die Menge der linearen Isometrieklassen von (n, k) – Codes bijektiv zu

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta} S_n\Big) \backslash\backslash GF(q)^{k\times n,k}.$$

Anwendung von Folgerung 1.1.22 auf Seite 9 auf dieses semidirekte Produkt liefert das Ergebnis. □

3.2. Semilineare Isometrieklassen linearer Codes

Nun wollen wir die Generatormatrizen von semilinear isometrischen Codes analog als Gruppenoperation eines geeigneten, semidirekten Produkts schreiben.
Wie wir bereits bewiesen haben, ist die Gruppe aller semilinearen Isometrien von $GF(q)^n$ gleich dem folgenden semidirekten Produkt:

$$GF(q)^{*n} \rtimes \big(Gal(q) \times S_n\big)$$

wobei die Multiplikation gegeben ist durch

$$\big(\varphi;(\beta,\rho)\big)\cdot\big(\psi;(\alpha,\pi)\big) := \big(\varphi\psi_{(\beta,\rho)};(\beta\alpha,\rho\pi)\big)$$

mit

$$\psi_{(\beta,\rho)}(i) := \beta(\psi(\rho^{-1}(i))), \quad i \in n$$

und

$$(\varphi\psi)(i) = \varphi(i)\psi(i), \quad i \in n.$$

Das Einselement dieser Gruppe ist $\big(1_n;(\tau^0,\mathrm{id})\big)$ und das Inverse eines Gruppenelements

$$\big(\psi;(\alpha,\pi)\big)^{-1} = \big(\psi^{-1}_{(\alpha^{-1},\pi^{-1})};(\alpha^{-1},\pi^{-1})\big)$$

mit

$$\psi^{-1}(i) := (\psi(i))^{-1}, \forall i \in n \text{ und } \psi^{-1}_{(\alpha^{-1},\pi^{-1})} := \big(\psi_{(\alpha^{-1},\pi^{-1})}\big)^{-1} = \big(\psi^{-1}\big)_{(\alpha^{-1},\pi^{-1})}.$$

3.2.1 Hilfssatz. *Diese Gruppe operiert auf den $k \times n$–Matrizen durch*

$$\Big(GF(q)^{*n} \rtimes \big(Gal(q)\times S_n\big)\Big)\times GF(q)^{k\times n} \to GF(q)^{k\times n}$$
$$\Big(\big(\psi;(\alpha,\pi)\big),\Gamma\Big) \mapsto \alpha(\Gamma)\cdot M^T_{(\psi;\pi)}$$

Beweis. Seien $\big(\varphi;(\beta,\rho)\big),\big(\psi;(\alpha,\pi)\big) \in GF(q)^{*n} \rtimes \big(Gal(q)\times S_n\big)$ beliebig gewählt und $\Gamma \in GF(q)^{k\times n}$, wir rechnen die Voraussetzung für eine Gruppenoperation nach:

$$\big(1_n;(\tau^0,\mathrm{id})\big)\Gamma = \tau^0(\Gamma)\cdot M^T_{(1_n;\mathrm{id})} = \Gamma$$

und

$$\begin{aligned}
\Big(\big(\varphi;(\beta,\rho)\big)\big(\psi;(\alpha,\pi)\big)\Big)\Gamma &= \Big(\big(\varphi\psi_{(\beta,\rho)};(\beta\alpha,\rho\pi)\big)\Big)\Gamma\\
&= (\beta\alpha)(\Gamma)\cdot M^T_{(\varphi\psi_{(\beta,\rho)};\rho\pi)}\\
&= \beta(\alpha(\Gamma))\cdot M^T_{(\varphi(\beta(\psi))_\rho;\rho\pi)}\\
&= \beta(\alpha(\Gamma))\cdot M^T_{(\beta(\psi);\pi)}\cdot M^T_{(\varphi;\rho)}\\
&= \beta(\alpha(\Gamma))\cdot \beta(M^T_{(\psi;\pi)})\cdot M^T_{(\varphi;\rho)}\\
&= \beta(\alpha(\Gamma)\cdot M^T_{(\psi;\pi)})\cdot M^T_{(\varphi;\rho)}\\
&= \beta\Big(\big(\psi;(\alpha,\pi)\big)\Gamma\Big)\cdot M^T_{(\varphi;\rho)}\\
&= \big(\varphi;(\beta,\rho)\big)\Big(\big(\psi;(\alpha,\pi)\big)\Gamma\Big)
\end{aligned}$$

□

Nun müssen wir nur noch die Operation der invertierbaren Matrizen von links berücksichtigen, um verschiedene Darstellungen eines Codes zusammenzuführen. Im Allgemeinen kommutiert diese Operation aber nicht mit der Operation der semilinearen Isometrien, was folgende Rechnung beweist:
Sei $A \in GL_k(q), \Gamma \in GF(q)^{k\times n}$ und $\big(\psi;(\alpha,\pi)\big) \in GF(q)^{*n} \rtimes \big(Gal(q)\times S_n\big)$ beliebig. Einerseits ergibt sich

$$A\cdot\big(\psi;(\alpha,\pi)\big)\Gamma = A\cdot\alpha(\Gamma)\cdot M^T_{(\psi;\pi)}$$

und andererseits

$$\big(\psi;(\alpha,\pi)\big)\big(A\cdot\Gamma\big) = \alpha(A\cdot\Gamma)\cdot M^T_{(\psi;\pi)} = \alpha(A)\cdot\alpha(\Gamma)\cdot M^T_{(\psi;\pi)}.$$

Wir müssen also bei der Multiplikation der $k\times k$–Matrix den Körperautomorphismus berücksichtigen.

3.2.2 Hilfssatz. $\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)$ *mit*

$$\theta : Gal(q)\times S_n \to Aut(GL_k(q)\times GF(q)^{*n}), (\alpha,\pi)\mapsto\theta((\alpha,\pi))$$

und

$$\theta((\alpha,\pi))(A,\psi) := (\alpha(A),\psi_{(\alpha,\pi)})$$

ist eine Gruppe, die auf der Menge $GF(q)^{k\times n}$ *durch*

$$\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)\times GF(q)^{k\times n}\to GF(q)^{k\times n}$$
$$\Big(\big((A,\psi);(\alpha,\pi)\big),\Gamma\Big)\mapsto A\cdot\alpha(\Gamma)\cdot M^T_{(\psi;\pi)}$$

operiert.

Beweis. Die Wohldefiniertheit und Homomorphieeigenschaft von θ kann leicht nachgerechnet werden. Es bleibt die Operation dieser Gruppe auf $GF(q)^{k\times n}$:

$$\big((I_k,1_n);(\tau^0,\mathrm{id}_{S_n})\big)\Big)\Gamma = \Gamma$$

und

$$\begin{aligned}
&\Big(\big((B,\varphi);(\beta,\rho)\big)\big((A,\psi);(\alpha,\pi)\big)\Big)\Gamma\\
=&\big((B\beta(A),\varphi\psi_{(\beta,\rho)});(\beta\alpha,\rho\pi)\big)\Gamma\\
=&B\beta(A)\cdot(\beta\alpha)(\Gamma)\cdot M^T_{(\varphi\psi_{(\beta,\rho)};\rho\pi)}\\
=&B\beta(A\cdot\alpha(\Gamma))\cdot\beta(M^T_{(\psi;\pi)})\cdot M^T_{(\varphi;\rho)}\\
=&B\beta\big(A\cdot\alpha(\Gamma)\cdot M^T_{(\psi;\pi)}\big)\cdot M^T_{(\varphi;\rho)}\\
=&\big((B,\varphi);(\beta,\rho)\big)\Big(\big((A,\psi);(\alpha,\pi)\big)\Gamma\Big)
\end{aligned}$$

□

3.2.3 Satz. *Die Menge der semilinearen Isometrieklassen von (n,k) – Codes ist bijektiv zu der Bahnenmenge*

$$\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)\backslash\backslash GF(q)^{k\times n,k}.$$

Ist C ein (n,k) – Code, so ist die Menge aller Generatormatrizen, die einen zu C semilinear isometrischen Code erzeugen gleich:

$$\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)(\Gamma)$$

für eine beliebige Generatormatrix Γ von C.

Beweis. Sei $\mathcal{U}(n,k,q) := \{V \leq GF(q)^n \mid \dim(V) = k\}$ die Menge aller (n,k) – Codes in $GF(q)^n$. Es sei $C(\Gamma)$ der von Γ erzeugte lineare Code. Wir zeigen, dass die Funktion

$$\begin{aligned}\Psi :&\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)\backslash\backslash GF(q)^{k\times n,k}\to\\ &\Big(GF(q)^{*n}\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)\backslash\backslash\mathcal{U}(n,k,q),\\ &\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)(\Gamma)\mapsto\\ &\Big(GF(q)^{*n}\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)(C(\Gamma))\end{aligned}$$

wohldefiniert und bijektiv ist. Ist

$$\omega\in\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)\backslash\backslash GF(q)^{k\times n,k}$$

beliebig, so gelten die folgenden Äquivalenzen, die zum einen die Wohldefiniertheit andererseits die Injektivität beweisen:

$$\begin{aligned}&\Gamma,\Gamma'\in\omega\\ \Longleftrightarrow&\exists\big((A,\varphi);(\alpha,\pi)\big)\in\Big(\big(GL_k(q)\times GF(q)^{*n}\big)\rtimes_\theta\big(Gal(q)\times S_n\big)\Big):\\ &A\cdot\alpha(\Gamma)\cdot M^T_{(\varphi;\pi)}=\Gamma'\\ \Longleftrightarrow&\exists\big(\varphi;(\alpha,\pi)\big)\in\Big(GF(q)^{*n}\rtimes\big(Gal(q)\times S_n\big)\Big):\\ &C\big(\alpha(\Gamma)\cdot M^T_{(\varphi;\pi)}\big)=C(\Gamma')\\ \Longleftrightarrow&\exists\big(\varphi;(\alpha,\pi)\big)\in\Big(GF(q)^{*n}\rtimes\big(Gal(q)\times S_n\big)\Big):\\ &\big(\varphi;(\alpha,\pi)\big)\ C(\Gamma)=C(\Gamma')\\ \Longleftrightarrow&\Big(GF(q)^{*n}\rtimes_\theta\big(Gal(q)\times S_n\big)\Big)(C(\Gamma))\\ &=\Big(GF(q)^{*n}\rtimes\big(Gal(q)\times S_n\big)\Big)(C(\Gamma'))\end{aligned}$$

Die Surjektivität ist klar, da es zu jedem Code eine Generatormatrix gibt. Die Bahn dieser Generatormatrix wird dann offensichtlich auf die Bahn des Codes abgebildet. □

3.2.4 Hilfssatz. *Sei $N \rtimes_\theta (G \times H)$ ein semidirektes Produkt mit einem direkten Produkt von Gruppen als rechter Seite. Es gilt:*

$$N \rtimes_\theta (G \times H) = (N \rtimes_{\theta_G} G) \rtimes_{\theta_H} H$$

mit $\theta_G : G \to Aut(N), g \mapsto \theta(g, 1_H)$
und $\theta_H : H \to Aut(N \rtimes_{\theta_G} G), h \mapsto \big(\theta(1_G, h), id_G\big)$

Beweis. Zunächst zeigen wir, dass die Funktionen θ_G und θ_H der Definition nach für ein semidirektes Produkt geeignet sind. Für θ_G ist dies offensichtlich erfüllt, da sie die Eigenschaften von θ erbt.
Sei nun $h, h' \in H$ und $(n, g), (n', g') \in N \rtimes_{\theta_G} G$ beliebig.

$$\begin{aligned}&\theta_H(hh')(n, g) = (\theta(1_G, hh')(n), g) = (\theta(1_G, h)(\theta(1_G, h')(n)), g)\\ &= \theta_H(h)(\theta(1_G, h')(n), g) = \theta_H(h)(\theta_H(h')(n, g)),\end{aligned}$$

also ist θ_H ein Homomorphismus. Wegen der Bijektivität von $\theta(1_G, h)$ und id_G ist die Abbildung $\theta_H(h)$ bijektiv. Bleibt die Homomorphieeigenschaft von $\theta_H(h)$:

$$\begin{aligned}&\theta_H(h)\big((n, g)(n', g')\big) = \theta_H(h)\big((n\theta_G(g)(n'), gg')\big) =\\ &= \big(\theta(1_G, h)(n\theta(g, 1_H)(n')), gg'\big)\\ &= \big(\theta(1_G, h)(n)\theta(g, h)(n'), gg'\big)\\ &= \big(\theta(1_G, h)(n)\theta(g, 1_H)(\theta(1_G, h)(n')), gg'\big)\\ &= \big(\theta(1_G, h)(n), g\big)\big(\theta(1_G, h)(n'), g'\big)\\ &= \theta_H(h)\big((n, g)\big)\ \theta_H(h)\big((n', g')\big)\end{aligned}$$

Bei beiden Gruppen ist die Grundmenge $N \times G \times H$, wir müssen nur noch nachprüfen, dass für beliebige Tripel $(n, g, h), (n', g', h') \in N \times G \times H$ die Multiplikationen analog definiert sind. Wir multiplizieren zunächst in $(N \rtimes_{\theta_G} G) \rtimes_{\theta_H} H$:

$$\begin{aligned}&(n, g, h)\ (n', g', h') = \big((n, g)\theta_H(h)(n', g'), hh'\big)\\ &= \big((n, g)(\theta(1_G, h)(n'), g'), hh'\big) = \big((n\theta_G(g)(\theta(1_G, h)(n')), gg'), hh'\big)\\ &= \big(n\theta(g, 1_H)(\theta(1_G, h)(n')), gg', hh'\big) = \big(n\theta(g, h)(n'), gg', hh'\big)\end{aligned}$$

Dies entspricht offensichtlich der Multiplikation beider Elemente in $N \rtimes_\theta (G \times H)$. □

3.2.5 Satz. *Die Menge der semilinearen Isometrieklassen von (n, k) – Codes ist bijektiv zu der Bahnenmenge*

$$S_n \backslash\backslash \Big[\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big) \backslash\backslash GF(q)^{k \times n, k}\Big]$$

mit

$$\begin{aligned}\theta_{Gal(q)} : Gal(q) &\to Aut\big(GL_k(q) \times GF(q)^{*n}\big)\\ \alpha &\mapsto \theta(\alpha, \mathrm{id}_{S_n})\end{aligned}$$

welches der komponentenweisen Anwendung des Körperautomorphismus α entspricht. Die Operationen sind wie folgt definiert:

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big) \times GF(q)^{k\times n,k} \to GF(q)^{k\times n,k}$$
$$\Big(\big((A,\varphi);\alpha\big),\Gamma\Big) \mapsto A \cdot \alpha(\Gamma) \cdot M^T_{(\varphi;\mathrm{id}_{S_n})}$$

und

$$S_n \times \Big[\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)\backslash\backslash GF(q)^{k\times n,k}\Big] \to$$
$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)\backslash\backslash GF(q)^{k\times n,k}$$
$$\Big(\pi, \Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)(\Gamma)\Big) \mapsto$$
$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)(\pi\Gamma)$$

Ist C ein (n,k) – Code, so ist die Menge aller Generatormatrizen, die einen zu C semilinear isometrischen Code erzeugen, gleich:

$$\bigcup_{\pi\in S_n} \Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)(\pi\Gamma)$$

für eine beliebige Generatormatrix Γ von C.

Beweis. Wir ersetzen nach obigem Hilfssatz die Definition der Gruppe

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_\theta \big(Gal(q) \times S_n\big)\Big)$$

durch die äquivalente Konstruktion

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big) \rtimes_{\theta_{S_n}} S_n.$$

Für dieses semidirekte Produkt ergibt Folgerung 1.1.22 die oben eingeführten Operationen des Normalteilers $\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)$ auf $GF(q)^{k\times n,k}$, sowie von S_n auf der Bahnenmenge dieser Operation.
Mit Hilfe der Folgerung erhalten wir weiterhin die behauptete Bijektivität der Bahnenmengen, sowie die Beschreibung aller Generatormatrizen der zu C semilinear isometrischen Codes. □

3.2.6 Bemerkung. Wir nennen die Operation von S_n auf

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)\backslash\backslash GF(q)^{k\times n,k}$$

die äußere Operation. Entsprechend bezeichnen wir die Operation von

$$\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes_{\theta_{Gal(q)}} Gal(q)\Big)$$

auf den $k \times n$–Matrizen mit vollem Zeilenrang als die innere Operation.

3.2.7 Bemerkung. Wir werden in der Notation von

$$\left(GL_k(q) \times GF(q)^{*n}\right) \rtimes_{\theta_{Gal(q)}} Gal(q)$$

den Homomorphismus $\theta_{Gal(q)}$ unterdrücken.

3.2.8 Bemerkung. Wir zeigen in Abschnitt 4.1, dass wir für die innere Operation auch ohne Backtrackingverfahren kanonische Repräsentanten berechnen können. Würden wir Algorithmus 2.8 direkt auf die Operation der Gruppe

$$\Big(\left(GL_k(q) \times GF(q)^{*n}\right) \rtimes_\theta \left(Gal(q) \times S_n\right)\Big) \leq S_{GF(q)^{k\times n}}$$

auf den $k \times n$-Matrizen anwenden, so hätten wir ein Baum mit bis zu

$$\left|\Big(\left(GL_k(q)\times GF(q)^{*n}\right) \rtimes_\theta \left(Gal(q)\times S_n\right)\Big)\right| = (q^k-1)\cdot(q^k-q)\cdots(q^k-q^{k-1})\cdot(q-1)^n\cdot r\cdot n!$$

Blättern zu durchsuchen. Im Gegensatz zu Leon [Leo82], der dieses Problem auf ein Backtrackverfahren in $S_{n\cdot(q-1)}$ überführt, erreichen wir mit dieser Formulierung eine weitesgehende[1] Unabhängigkeit von der Ordnung des Körpers.

Wie wir im Algorithmus geschickt mit der inneren Bahnenmenge umgehen, werden wir im nächsten Kapitel ausführlich beschreiben. Dort werden außerdem die notwendige Totalordnung und die benutzten Homomorphismen bereitgestellt. Wir wollen an dieser Stelle nun noch ausarbeiten, wie wir die Ausgabe des Algorithmus auf unser Ausgangsproblem zurückführen.

3.3. Rücktransformation der Ergebnisse

Wir beweisen die Sätze für ein beliebiges semidirektes Produkt $N \rtimes H$, welches auf einer Menge X operiere. Für die Ausgangssituation der Kanonisierung von Generatormatrizen linearer Codes brauchen wir dann nur $H = S_n$, $N = (GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)$ und $X = GF(q)^{k\times n,k}$ zu setzen.

3.3.1 Hilfssatz. *Sei $G := N \rtimes_\theta H$ ein semidirektes Produkt, welches auf einer Menge X operiere. Wir erhalten aus einer Transversale T_H der induzierten Operationen von H auf $N\backslash\backslash X$, siehe Folgerung 1.1.22, und einer Transversalen T_N von N auf X eine Transversale T_G der G–Bahnen auf X:*

$$T_G := \{x \in X \mid N(x) \in T_H \wedge x \in T_N\}$$

[1] siehe Bemerkung 4.1.26

Beweis. Wir wenden das Homomorphieprinzip, Satz 1.1.12, für die folgenden Abbildungen an:

$$\theta : X \to N\backslash\backslash X, \quad x \mapsto N(x),$$
$$\varphi : G \to H, \quad (n,h) \mapsto h.$$

Die notwendigen Voraussetzungen folgen aus Hilfssatz 1.1.21. Es gilt somit:

$$T_{G\backslash\backslash X} := \bigcup_{N(x)\in T_H} T_{\varphi^{-1}(H_{N(x)})\backslash\backslash N(x)}$$

ist eine Transversale von G auf X.

Da aber bereits $\varphi^{-1}(\{1_H\}) = N \rtimes \{1_H\} \simeq N$ ist, folgt insbesondere

$$\varphi^{-1}(H_{N(x)}) \geq N \rtimes \{1_H\}.$$

Die Operation von N ist aber gerade über die isomorphe Untergruppe definiert, somit besteht die Transversale $T_{\varphi^{-1}(H_{N(x)})\backslash\backslash N(x)}$ nur aus einem Element. Wir können dieses aus $N(x)$ beliebig wählen und definieren es wie folgt

$$T_{\varphi^{-1}(H_{N(x)})\backslash\backslash N(x)} := \{x' \in N(x) \mid x' \in T_N\}.$$

Somit gilt aber $T_{G\backslash\backslash X} = T_G$. □

3.3.2 Hilfssatz. *Es sei $x \in X$ beliebig. Seien die Voraussetzungen analog zu oben und E_N ein Erzeugendensystem von N_x, sowie E_H ein Erzeugendensystem von $H_{N(x)}$. Desweiteren sei zu $h \in E_H$ ein $n_h \in N$ mit $(n_h, h)x = x$ berechnet. Dann ist*

$$E := \{(n_h, h) \mid h \in E_H\} \cup \{(n, 1_H) \mid n \in E_N\}$$

ein Erzeugendensystem von G_x.

Beweis. Es ist $E \subseteq G_x$, wir brauchen also nur noch zu zeigen, dass die Menge auch den Stabilisator erzeugt. Wir wählen $(n,h) \in G_x$ beliebig:

$$\begin{aligned}
&x = (n,h)x = (n,1_H)(1_N,h)x \\
\Rightarrow &N(x) = N(hx) = hN(x) \\
\Rightarrow &h \in H_{N(x)} \\
\Rightarrow &\exists r \in \mathbb{N}, h_0, \ldots, h_{r-1} \in E_H \text{ sd. } h = h_0 \cdots h_{r-1} \\
\Rightarrow &(n_{h_0}, h_0) \cdots (n_{h_{r-1}}, h_{r-1}) =: (\widetilde{n}, h) \in \langle E \rangle
\end{aligned}$$

weiter ist

$$\begin{aligned}
&(\widetilde{n}, h)^{-1}(n,h) =: (m, 1_H) \in G_x \\
\Rightarrow &m \in N_x \\
\Rightarrow &(m, 1_H) \in \langle E \rangle \\
\Rightarrow &(\widetilde{n}, h)(m, 1_H) = (n,h) \in \langle E \rangle
\end{aligned}$$

□

3.3.3 Folgerung. *Es sei Can_H ein System kanonischer Transversalen von H auf $N\backslash\backslash X$ und can_H ein zugehöriger Kanonisierer. Weiterhin sei noch γ eine kanonisierende Abbildung für eine Transversale T_N der Operation von N auf X. Dann ist die Abbildung*

$$\mu : X \to G, x \mapsto (\gamma(h_x x), h_x)$$

mit $(h_x, H_{N(x)}) := can_H(H, N(x))$ eine G–kanonisierende Abbildung bezüglich der Transversalen T_G.

Beweis. Es gilt für ein beliebiges $x \in X$

$$N\big((\gamma(h_x x), h_x)x\big) = N\big((\gamma(h_x x), 1_H)(1_N, h_x)x\big) = N(h_x x) = h_x N(x) \in Can(H)$$

und $(\gamma(h_x x), h_x)x \in T_N$. Damit ist aber $\mu(x)x = (\gamma(h_x x), h_x)x \in T_G$ und μ somit G–kanonisierend. □

4. Die äußere Kanonisierung

Um nun einen Algorithmus zur Bestimmung eindeutiger Repräsentanten der semilinearen Isometrieklassen eines linearen Codes angeben zu können, ist es nach Kapitel 3 nur noch nötig, einen Algorithmus zur Kanonisierung für die Operation der Gruppe S_n auf $\Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k}$ zur Verfügung zu stellen. Der Algorithmus 2.8 gibt uns hierfür den Rahmen. Wir müssen im Wesentlichen nur geeignete Homomorphismen für jede Ebene zur Verfügung stellen. Die Stabilisatorkette werde zur Basis $\overrightarrow{b} = (0, \ldots, n-1)$ gebildet. Desweiteren werden wir, da die Menge X in unserem Fall sehr kompliziert erscheint, genau beschreiben, wie diese verwaltet wird. Auch geben wir einen Hinweis darauf, wie die Transversalenelemente $T_i^{(j)}, i, j \in n, j \geq i$ der betrachteten Untergruppen günstig bestimmt werden können. In diesem Kapitel gehen wir stets davon aus, dass wir eine Generatormatrix $\Gamma \in GF(q)^{k\times n,k}$ ohne Nullspalten zur Kanonisierung vorliegen haben. Dass dies auch eine berechtigte Annahme ist, zeigen wir in Abschnitt 4.3.1.

4.1. Verwaltung der inneren Bahnen

Aus Gründen der Übersichtlichkeit führen wir folgende, abkürzende Schreibweise für die Bahnen der inneren Operation ein.

4.1.1 Definition. Es sei $\Gamma \in GF(q)^{k\times n,k}$ beliebig:

$$\omega(\Gamma) := \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big)(\Gamma)$$

4.1.2 Bemerkung. Sind $g, h \in S_n$ und $\Gamma \in GF(q)^{k\times n,k}$ beliebig, so gilt: $g\omega(h\Gamma) = \omega(gh\Gamma)$.

Als weitere Voraussetzung zum Starten des Algorithmus 2.8 benötigen wir noch eine lineare Ordnung auf

$$\Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k}$$

und definieren:

4.1.3 Definition. Es seien $\omega_1, \omega_2 \in \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k}$ beliebig.

$$\omega_1 \leq \omega_2 :\Longleftrightarrow \min\{A \in \omega_1\} \leq \min\{A \in \omega_2\},$$

wobei die Minima bezüglich einer beliebigen Totalordnung auf den $k \times n$–Matrizen gebildet werden. Wir vergleichen also die inneren Bahnen mittels ihrer eindeutigen Repräsentanten in $Can_{min}\Big(\big(GL_k(q) \times GF(q)^{*^n}\big) \rtimes Gal(q)\Big)$.

Bei näherer Betrachtung der Ablaufreihenfolge des Algorithmus zeigt sich, dass nicht jede Totalordnung der $k \times n$–Matrizen geeignet ist, um frühestmöglich Teilbäume abschneiden zu können. Da der zugrundeliegende Körper im Algorithmus beliebig implementiert sein kann, setzen wir an dieser Stelle nur voraus, dass eine Totalordnung $(GF(q), \leq)$ gegeben sei, mit

$$0 < 1 < x, \forall x \in GF(q) \setminus \{0, 1\}.$$

4.1.4 Definition (Lexikographische Ordnung)**.** Es sei $(X, \leq)$ eine totalgeordnete Menge. Wir definieren für $x, y \in X^n$ die Relation:

$$x \leq y :\Longleftrightarrow x = y \vee \big(\exists i \in n : x_i < y_i \wedge x_j = y_j \text{ für } 0 \leq j < i\big)$$

Durch diese Relation erhalten wir die lexikographische Ordnung – eine Totalordnung – auf X^n.
Wir können die lexikographische Sortierung auch von rechts nach links durchführen:

$$x \leq y :\Longleftrightarrow x = y \vee \big(\exists i \in n : x_i < y_i \wedge x_j = y_j \text{ für } i < j \leq n-1\big),$$

hierdurch erhalten wir die revers lexikographische Ordnung, eine weitere Totalordnung auf X^n.

Aus der revers lexikographischen Ordnung gewinnen wir zunächst eine Totalordnung auf den k–dimensionalen Vektorraum $GF(q)^k$. Schließlich erhalten wir weiter durch den lexikographischen Vergleich der Spalten eine lineare Anordnung auf $GF(q)^{k \times n}$.

Wir werden nicht bei jeder Multiplikation eines Transversalenelements $\overline{T}^{(i-1)}[j]$ bereits den kanonischen Repräsentanten bezüglich Can_{min} bestimmen um die innere Bahn verwalten zu können. Vielmehr nutzen wir die folgende Beobachtung:
Es sei g_i ein Knoten auf Ebene i. Für eine Permutation $\pi \in S_n^{(i)}$ gilt:

$$\begin{aligned}\pi g_i \Gamma &= \big(\Gamma(g_i^{-1}\pi^{-1}(0))^T\big|\ldots\big|\Gamma(g_i^{-1}\pi^{-1}(n-1))^T\big)\\ &= \big(\Gamma(g_i^{-1}(0))^T\big|\ldots\big|\Gamma(g_i^{-1}(i-1))^T\big|\Gamma(g_i^{-1}\pi^{-1}(i))^T\big|\ldots\big|\Gamma(g_i^{-1}\pi^{-1}(n-1))^T\big)\end{aligned}$$

Die ersten i Spalten bleiben also fix. Beim Übergang auf Ebene $i+1$ wählen wir ein Transversalenelement $\overline{T}^{(i)}[j]$, was nichts anderes bedeutet, als dass wir die Spalte $\Big(\overline{T}^{(i)}[j]\Big)^{-1}(i)$ wählen und an Position i verschieben. Im darunterliegenden Teilbaum wird diese Spalte stets an der i–ten Stelle belassen.

4.1.5 Bemerkung. Diese Beobachtung zeigt, warum für unsere Zwecke die Durchlaufreihenfolge entlang der Linkstransversalen der Stabilisatorkette von Vorteil ist. Wir gewinnen nicht nur den Test mittels Hilfssatz 2.4.9, sondern können auch die Informationen des Vorgängers vollständig ausnutzen, d.h. wir brauchen keine erneute Minimierung des Repräsentanten starten.

4.1.6 Definition (i–te Projektion)**.** Sei $i \in (n+1)$ gegeben. Die Abbildung

$$\begin{aligned}\Pi_i : GF(q)^{k\times n} &\to GF(q)^{k\times i},\\ \big(A(0)^T\big|\ldots\big|A(n-1)^T\big) &\mapsto \big(A(0)^T\big|\ldots\big|A(i-1)^T\big)\end{aligned}$$

heißt i–te Projektion der Matrix A.

4.1.7 Folgerung. *Es sei $g \in S_n$ ein beliebige Permutation. Die i–te Projektion der Menge $\omega(g\Gamma)$ bleibt fix für alle $h \in S_n^{(i)}$:*

$$h \in S_n^{(i)} \Rightarrow \Pi_i(\omega(g\Gamma)) = \Pi_i(\omega(hg\Gamma)).$$

4.1.8 Definition. Wir nennen eine Matrix Γ genau dann i–minimal in ihrer Bahn $\omega(\Gamma)$, falls die i–te Projektion von Γ minimal ist in der Menge $\Pi_i\big(\omega(\Gamma)\big)$:

$$\Gamma \text{ } i\text{–minimal} :\Longleftrightarrow \forall\widetilde{\Gamma} \in \omega(\Gamma) : \Pi_i(\Gamma) \leq \Pi_i(\widetilde{\Gamma}).$$

Um nun die inneren Bahnen $\omega(g_i\Gamma)$ auf Ebene i während des Backtrackverfahrens besser handhaben zu können, wählen wir stets einen i–minimalen Repräsentanten $\Gamma^{(g_i,i)}$ als Vertreter der Bahn.

4.1.9 Folgerung. *Da $\Pi_n = \mathrm{id}$ folgt, dass $\Gamma^{(g_n,n)}$ minimal in $\omega(g_n\Gamma)$ ist und somit der eindeutige Vertreter der Transversalen*

$$Can_{min}\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes Gal(q)\Big).$$

4.1.10 Folgerung. *Wählen wir als Operation von $S_n^{(i)}$ auf $GF(q)^{k\times i}$ die triviale Definition $(\pi, A) \mapsto A$, so ist die Abbildung*

$$\omega(g_i\Gamma) \mapsto \Pi_i\big(\Gamma^{(g_i,i)}\big)$$

ein $S_n^{(i)}$–Homomorphismus. Insbesondere können wir diesen zum frühzeitigen Abschneiden von Teilbäumen benutzen.

4.1.11 Bemerkung. Seien $g \in S_n$ und $i \in (n+1)$ beliebig. Aus der spaltenweise lexikographischen Sortierung der Matrizen folgt für alle $0 \leq j \leq i$:

$$\Pi_j\big(\Gamma^{(g,i)}\big) = \Pi_j\big(\Gamma^{(g,j)}\big) \text{ sowie } \forall h \in S_n^{(i)} : \Pi_j\big(\Gamma^{(hg,j)}\big) = \Pi_j\big(\Gamma^{(g,j)}\big).$$

4.1.12 Hilfssatz. *Die Gruppe* $(GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)$ *operiert durch*

$$\begin{aligned}\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big) \times GF(q)^{k\times i} &\to GF(q)^{k\times i}\\ \big(((A,\varphi);\alpha),\Gamma\big) &\mapsto ((A,\varphi\downarrow_i);\alpha)\Gamma\end{aligned}$$

auch auf der Menge $GF(q)^{k\times i}$ *und die Abbildung* Π_i *ist ein* $\big((GL_k(q)\times GF(q)^{*n})\rtimes Gal(q)\big)$ *– Homomorphismus.*

Beweis. Einfaches Nachrechnen der Definitionen. □

Zur rekursiven Berechnung eines $(i+1)$ – minimalen Repräsentanten $\Gamma^{(g_{i+1},i+1)}$ im Backtrackdurchlauf aus einem i–minimalen Repräsentanten $\Gamma^{(g_i,i)}$ des Vaterknotens müssen wir also die Spalte $\left(\overline{T}^{(i-1)}[j]\right)^{-1}(i)$ von $\Gamma^{(g_i,i)}$ bezüglich des Stabilisators $\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(g_i,i)})}$ der ersten i Spalten von $\Gamma^{(g_i,i)}$ minimieren. Offensichtlich gilt weiterhin nach obiger Bemerkung:

$$\big((GL_k(q)\times GF(q)^{*n})\rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(g_{i+1},i+1)})} = \big((GL_k(q)\times GF(q)^{*n})\rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(g_i,i)})}.$$

Wir wollen nun beschreiben, wie wir den Stabilisator

$$\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(g_i,i)})}$$

rekursiv berechnen, sowie im Backtrackverfahren effizient abspeichern können.

4.1.13 Definition. Es sei t ein Teiler von r, $s \in k$, $i \in (n+1)$ und $\mathfrak{p} = \{p_0,\dots,p_{l-1}\}$ eine Partition der Menge $\{0,\dots,s-1\}$. Weiter sei noch $\Gamma \in GF(q)^{k\times n}$ beliebig. Wir definieren die folgenden Teilmengen von $(GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)$:

$$\begin{aligned}\mathcal{A}(t) &:= \big\{((I_k,1_n);\tau^{tx}) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\}\\ \mathcal{G}(s) &:= \left\{\left(\left(\begin{pmatrix} I_s & A^{(1)}\\ 0 & A^{(2)}\end{pmatrix},1_n\right);\tau^0\right)\,\middle|\, A^{(1)}\in GF(q)^{s\times k-s}, A^{(2)}\in GL_{k-s}(q)\right\}\\ \mathcal{C}(i) &:= \big\{((I_k,\varphi);\tau^0)\big|\varphi\in GF(q)^{*n}, \forall j<i : \varphi(j)=1\big\}.\\ \mathcal{Z}(p_j,\Gamma) &:= \left\{((D,\varphi);\tau^0)\,\middle|\, \exists\mu\in GF(q)^* \text{ sd. } D=\operatorname{diag}(d_0,\dots,d_{k-1}),\right.\\ &\qquad d_p=\begin{cases}\mu, p\in p_j\\ 1,\text{sonst}\end{cases}, \varphi(s)=\left.\begin{cases}\mu^{-1},\operatorname{supp}(\Gamma(s))\subseteq p_j\\ 1,\text{sonst}\end{cases}\right\}\\ \mathcal{E}(i,s,\mathfrak{p},t,\Gamma) &:= \mathcal{A}(t)\cup\mathcal{G}(s)\cup\mathcal{C}(i)\cup\bigcup_{j\in l}\mathcal{Z}(p_j,\Gamma)\end{aligned}$$

4.1.14 Bemerkung. Wir können die einzelnen Mengen, wie folgt interpretieren:

$\mathcal{A}(t)$ Untergruppe der Körperautomorphismen.

$\mathcal{G}(s)$ beliebige Gaußschritte auf den Zeilen, welche die ersten s Einheitsvektoren[1] erhalten, dh. Multiplikation einer Zeile mit Index $\geq s$ mit einem Element aus $GF(q)^*$, Vertauschung von zwei Zeilen mit Indizes $\geq s$, Addition einer Zeile mit Index $\geq s$ zu einer beliebigen anderen.

$\mathcal{C}(i)$ Beliebige Multiplikation von Spalten $\geq i$ mit Elementen aus $GF(q)^*$.

$\mathcal{Z}(p_j, \Gamma)$ Gekoppelte Zeilen-/Spaltenmultiplikation, dh. Multiplikation aller Zeilen in p_j mit $\mu \in GF(q)^*$ und Ausgleich dieser Multiplikation mit Spaltenmultiplikation μ^{-1} für diejenigen Spalten deren Träger supp(Γ) komplett in p_j liegt.

4.1.15 Hilfssatz. *Sei* $((A, \varphi); \alpha) \in \langle \mathcal{E}(i, s, \mathfrak{p}, t, \Gamma) \rangle$*, dann gibt es eine Darstellung der Form*

$$((A, \varphi); \alpha) = g \cdot \prod_{j \in l} z_j \cdot c \cdot a$$

mit $g \in \mathcal{G}(s)$*,* $z_j \in \mathcal{Z}(p_j, \Gamma)$*,* $c \in \mathcal{C}(i)$*,* $a \in \mathcal{A}(t)$*.*

Beweis. Wir haben für jedes Paar $\mathcal{X} \neq \mathcal{Y}$ der obigen Mengen zu zeigen, dass die Multiplikation ihrer Elemente auch in umgekehrter Reihenfolge erfolgen kann:

$$\forall x \in \mathcal{X}, y \in \mathcal{Y} : \exists x' \in \mathcal{X}, y' \in \mathcal{Y} \text{ sd. } xy = y'x'.$$

Desweiteren gilt für die Mengen $\mathcal{X}$, dass sie unter Multiplikation abgeschlossen sind:

$$\forall x, x' \in \mathcal{X} : \exists \overline{x} \text{ sd. } xx' = \overline{x}.$$

Diese Eigenschaften sind leicht nachzurechnen, sie folgen aus der Kommutativität des Körpers, den Eigenschaften des semidirekten Produkts und des direkten Produkts von Gruppen, sowie der Multiplikation von Diagonalmatrizen. Aus diesem Grunde wird der Beweis an dieser Stelle nicht geführt.
Wählen wir dann ein beliebiges Element aus dem Erzeugnis, so können wir aus einer Darstellung mit Erzeugern durch sukzessives, geeignetes Vertauschen und Zusammenfassen die obige Darstellung erreichen. □

4.1.16 Folgerung. *Jede, der in Definition 4.1.13 eingeführten Mengen mit Ausnahme von* $\mathcal{E}(i, s, \mathfrak{p}, t, \Gamma)$*, ist eine Untergruppe von* $(GL_k(q) \times GF(q)^{*^n}) \rtimes Gal(q)$*.*

Beweis. Die unter Multiplikation abgeschlossenen Teilmengen einer endlichen Gruppe sind Untergruppen. Die Gruppe $(GL_k(q) \times GF(q)^{*^n}) \rtimes Gal(q)$ ist endlich, und nach obigem Hilfssatz sind diese Teilmengen unter Multiplikation abgeschlossen. □

[1] als Spalten

4.1.17 Definition. Wir nennen zu $i \in (n+1)$ und $\Gamma^{(g_i,i)}$ das Tripel $(s, \mathfrak{p}, t)$ mit

- $\mathrm{Rang}\big(\Pi_i(\Gamma^{(g_i,i)})\big) = s$,
- $\mathfrak{p} = \{p_0, \ldots, p_{l-1}\}$ Partition der Menge $\{0, \ldots, s-1\}$ und
- $t|r$

genau dann Stabilisatortripel der ersten i Spalten von $\Gamma^{(g_i,i)}$, falls die Menge $\mathcal{E}(i, s, \mathfrak{p}, t, \Gamma^{(g_i,i)})$ ein Erzeugendensystem der Gruppe

$$\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(g_i,i)})}$$

ist und falls weiter t minimal mit dieser Eigenschaft ist.

4.1.18 Hilfssatz. *Die Parameter $(s, \mathfrak{p}, t)$ eines Stabilisatortripels zu den ersten i Spalten sind gleich für jeden i–minimalen Repräsentanten $\Gamma^{(g_i,i)}$ der Bahn $\omega(\Gamma)$.*

Beweis. Es sei $(s', \mathfrak{p}', t')$ ein Stabilisatortripel zu den ersten i Spalten eines weiteren i–minimalen Repräsentanten Γ'.
Da die ersten i Spalten wegen der i–Minimalität gleich sind, ist

$$s = s' = \mathrm{Rang}\big(\Pi_i(\Gamma^{(g_i,i)})\big).$$

Weiter ist

$$\big\langle \mathcal{E}(i, s, \mathfrak{p}, t, \Gamma^{(g_i,i)}) \big\rangle = \big\langle \mathcal{E}(i, s', \mathfrak{p}', t', \Gamma') \big\rangle$$

und somit müssen die Untergruppen $\mathcal{A}(t)$ und $\mathcal{A}(t')$ gleich sein. Weil aber t bzw. t' als minimal vorausgesetzt wurden, gilt $t = t'$.
Wir müssen also nur noch die Gleichheit der Mengenpartitionen zeigen. Hierzu wählen wir ein beliebiges $p' \in \mathfrak{p}'$ und stellen ein nichttriviales Element $z' \in \mathcal{Z}(p', \Gamma')$ mit Hilfssatz 4.1.15 durch Erzeuger in $\mathcal{E}(i, s, \mathfrak{p}, t, \Gamma^{(g_i,i)})$ dar:

$$z' = g \cdot \prod_{j \in l} z_j \cdot c \cdot a.$$

Die Beobachtung der Matrixkomponenten und des Körperautomorphismus ergibt, dass sowohl g als auch a das Einselement der Gruppe sein müssen. Es bleibt:

$$z' = \prod_{j \in l} z_j \cdot c.$$

Da die Zeilenmultiplikationen z_j aber auf disjunkte Zeilenindexmengen wirken und c nur die Einheitsmatrix enthält, folgt für eine eindeutige Indexmenge $I \subseteq l$:

$$p' = \dot{\bigcup}_{l \in I} p_l.$$

Aus Symmetriegründen folgt auch die Umkehrung und somit ist $\mathfrak{p} = \mathfrak{p}'$. □

4.1.19 Folgerung. *Die Parameter $(s, \mathfrak{p}, t)$ eines Stabilisatortripels zu den ersten i Spalten von $\Gamma^{(g_i,i)}$ sind eindeutig.*

4.1.20 Hilfssatz. *Ist $(s, \mathfrak{p}, t)$ das Stabilisatortripel zu den ersten i Spalten von $\Gamma^{(g_i,i)}$ und $h \in S_n^{(i)}$, so ist $(s, \mathfrak{p}, t)$ auch das Stabilisatortripel zu dem i–minimalen Repräsentanten $h \cdot \Gamma^{(g_i,i)}$ der Bahn $\omega(hg_i\Gamma)$.*

Beweis. Es unterscheiden sich zwar die Elemente der Mengen $\mathcal{Z}(p_j, \Gamma^{(g_i,i)})$ und $\mathcal{Z}(p_j, h \cdot \Gamma^{(g_i,i)})$, jedoch nur in der Multiplikation von verschiedenen Spalten mit Index $\geq i$. Da aber in beiden Erzeugendensystemen die Mengen $\mathcal{C}(i)$ enthalten sind, können wir dies durch entsprechende Multiplikation der Spalten ausgleichen. Da außerdem $\Pi_i(\Gamma^{(g_i,i)}) = \Pi_i(h \cdot \Gamma^{(g_i,i)})$ gilt, ist auch $(s, \mathfrak{p}, t)$ das Stabilisatortripel der ersten i Spalten von $h \cdot \Gamma^{(g_i,i)}$. □

4.1.21 Satz. *Zu jedem $i \in (n+1)$ und jedem i–minimalen Repräsentanten $\Gamma^{(g_i,i)}$ der Bahn $\omega(g_i\Gamma)$ gibt es ein Stabilisatortripel.*

Beweis. Folgt aus dem restlichen Abschnitt durch Induktion nach n. □

4.1.22 Hilfssatz. $\mathcal{E}(0, 0, \{\emptyset\}, 1, \Gamma)$ *ist ein Erzeugendensystem von*

$$\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_0(\Gamma)} = (GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q).$$

Beweis. Sei $((A, \varphi); \alpha) \in (GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)$ beliebig. Es gilt:

$$((A, \varphi); \alpha) = \underbrace{((A, 1_n); \tau^0)}_{\in \mathcal{G}(0)} \underbrace{((I_k, \varphi); \tau^0)}_{\in \mathcal{C}(0)} \underbrace{((I_k, 1_n); \tau^{1 \cdot m})}_{\in \mathcal{A}(1)}$$

mit $1 \leq m \leq r$ und $\tau^m = \alpha$. □

Wir wählen also auf Ebene 0 des Backtrackbaums Γ als 0–minimalen Repräsentanten und $(0, \{\emptyset\}, 1)$ als Stabilisatortripel der ersten 0 Spalten von Γ.

Nun geben wir an, wie wir die Berechnung auf Ebene $i+1, i \in n$ aus der Information des Vaters g_i mit i–minimalem Repräsentanten $\Gamma^{(g_i,i)}$ und Stabilisatortripel $(s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)})$ durchführen. Hierzu haben wir zwei Fälle zu unterscheiden, je nachdem ob die nächste Spalte $(g_{i+1}\Gamma)(i)^T$ im Erzeugnis der vorausgegangenen Spalten

$$\left\langle \Gamma^{(g_i,i)}(0)^T, \ldots, \Gamma^{(g_i,i)}(i-1)^T \right\rangle$$

liegt, oder nicht. Da $g_{i+1} = hg_i$ für ein $h \in S_n^{(i)}$ können wir nach Hilfssatz 4.1.20 ohne Beschränkung der Allgemeinheit voraussetzen, dass $g_{i+1} = g_i$ gilt. Wir minimieren dann die Spalte $\Gamma^{(g_i,i)}(i)^T$ unter dem Stabilisator der vorausgegangen Spalten

$$\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(g_i,i)})}.$$

Weiter zeigt die induktive Anwendung der nachfolgenden Hilfssätze, dass $\Gamma^{(g_i,i)}$ die Einheitsvektoren $e_0^T, \ldots, e_{s-1}^T, s := \mathrm{Rang}(\Pi_i(\Gamma^{(g_i,i)}))$ als Spalten enthält.

4.1.23 Hilfssatz. *Sei $(s, \mathfrak{p}, t)$ das Stabilisatortripel der ersten i Spalten von $\Gamma^{(g_i,i)}$. Desweiteren gelte:*

$$\Gamma^{(g_i,i)}(i)^T \notin \left\langle \Gamma^{(g_i,i)}(0)^T, \ldots, \Gamma^{(g_i,i)}(i-1)^T \right\rangle.$$

In $\omega(g_i\Gamma)$ existiert dann eine Matrix $\Gamma^{(g_i,i+1)}$ mit

$$\Pi_{i+1}\left(\Gamma^{(g_i,i+1)}\right) = \left(\Gamma^{(g_i,i)}(0)^T \middle| \ldots \middle| \Gamma^{(g_i,i)}(i-1)^T \middle| e_s^T\right).$$

Wir können diese Matrix als $(i+1)$ – minimalen Repräsentanten der Bahn $\omega(g_i\Gamma)$ wählen. Das Tripel $(s+1, \mathfrak{p} \cup \{\{s\}\}, t)$ ist das Stabilisatortripel der ersten $i+1$ Spalten von $\Gamma^{(g_i,i+1)}$.

Beweis. Es sei $\Gamma^{(g_i,i)}(i)^T = (\gamma_0, \ldots, \gamma_{k-1})^T$ die zu minimierende Spalte. Da die Spalte $\Gamma^{(g_i,i)}(i)^T$ nicht im Erzeugnis der Vorausgegangenen liegt, diese aber die ersten s Einheitsvektoren nach Induktionsvoraussetzung umfassen, gibt es ein $s' \geq s$ mit $\gamma_{s'} \neq 0$. Die Vertauschung der Zeilen s und s' können durch das Gruppenelement $((A, 1_n); \tau^0) \in \mathcal{G}(s)$ erreicht werden, wobei die Matrix A aus der Einheitsmatrix durch Vertauschen der Zeilen s und s' hervorgeht.

Wir können also weiter annehmen, dass $\gamma_s \neq 0$ ist. Durch die Matrix

$$B := \begin{pmatrix} & & -\gamma_0\gamma_s^{-1} & & \\ & I_s & \vdots & 0 & \\ & & -\gamma_{s-1}\gamma_s^{-1} & & \\ 0 \ \ldots \ 0 & & \gamma_s^{-1} & 0 \ \ldots \ 0 & \\ & & -\gamma_{s+1}\gamma_s^{-1} & & \\ & 0 & \vdots & I_{k-s-1} & \\ & & -\gamma_{k-1}\gamma_s^{-1} & & \end{pmatrix}$$

bringen wir die i–te Spalte auf den Einheitsvektor e_s:

$$B\Gamma^{(g_i,i)}(i)^T = e_s^T.$$

Nun ist aber $((B, 1_n); \tau^0) \in \mathcal{G}(s)$ und somit

$$\Gamma^{(g_i,i+1)} := ((B, 1_n); \tau^0)\Gamma^{(g_i,i)} \in \omega(g_i\Gamma).$$

Desweiteren ist die Projektion von $\Gamma^{(g_i,i+1)}$ auf die ersten $i+1$ Spalten minimal in $\Pi_i(\omega(g_{i+1}\Gamma))$ und deshalb $\Gamma^{(g_i,i+1)}$ ein $(i+1)$ – minimaler Repräsentant für die innere Bahn.

Es bleibt die Behauptung für das Stabilisatortripel zu überprüfen:
Die Parameter $(s+1, \mathfrak{p} \cup \{\{s\}\}, t)$ sind nach Induktionsvoraussetzung zulässig gewählt.

Wir müssen nur noch zeigen, dass $\mathcal{E}(i+1, s+1, \mathfrak{p} \cup \{\{s\}\}, t, \Gamma^{(g_{i+1}, i+1)})$ ein Erzeuger der Gruppe $\left((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\right)_{\Pi_{i+1}(\Gamma^{(g_{i+1}, i+1)})}$ ist. Die Behauptung

$$\mathcal{E}(i+1, s+1, \mathfrak{p} \cup \{\{s\}\}, t, \Gamma^{(g_{i+1}, i+1)}) \subseteq \left((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\right)_{\Pi_{i+1}(\Gamma^{(g_{i+1}, i+1)})}$$

ist leicht nachzuprüfen, wir zeigen deshalb nur die Erzeugereigenschaft. Sei dazu

$$((A, \varphi); \alpha) \in \left((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\right)_{\Pi_{i+1}(\Gamma^{(g_{i+1}, i+1)})}$$

beliebig gewählt. Nach Hilfssatz 4.1.15 können wir das Gruppenelement durch Erzeuger aus $\mathcal{E}(i, s, \mathfrak{p}, t, \Gamma^{(g_i, i)})$, ähnlich wie dort angegeben[2], darstellen:

$$((A, \varphi); \alpha) = g \cdot c \cdot \prod_{j \in l} z_j \cdot a$$

mit $g \in \mathcal{G}(s)$, $z_j \in \mathcal{Z}(p_j, \Gamma^{(g_i, i)})$, $c \in \mathcal{C}(i)$, $a \in \mathcal{A}(t)$. Da weiter

$$a^{-1}, z_j^{-1} \in \left\langle \mathcal{E}(i+1, s+1, \mathfrak{p} \cup \{\{s\}\}, t, \Gamma^{(g_{i+1}, i+1)}) \right\rangle, \forall j \in l,$$

reicht es zu zeigen, dass auch das Produkt

$$g \cdot c \in \left((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\right)_{\Pi_{i+1}(\Gamma^{(g_{i+1}, i+1)})}$$

erzeugt werden kann.
Wir teilen die Spaltenmultiplikation $c = ((I_k, \varphi); \tau^0)$ wie folgt in

$$c = ((I_k, \varrho); \tau^0) \cdot ((I_k, \sigma); \tau^0) \text{ mit}$$
$$\varrho(j) := \begin{cases} \varphi(j), j = i \\ 1, j \neq i \end{cases} \quad \text{und} \quad \sigma(j) := \begin{cases} \varphi(j), j > i \\ 1, j \leq i \end{cases} .$$

Es gilt $((I_k, \varrho); \tau^0) \in \mathcal{C}(i)$ und $((I_k, \sigma); \tau^0) \in \mathcal{C}(i+1)$. Analog zu oben, können wir mit dem Inversen $((I_k, \sigma); \tau^0)^{-1}$ von rechts multiplizieren. Und somit bleibt nur noch für das Produkt $g \cdot ((I_k, \varrho); \tau^0)$ die Erzeugbarkeit mittels Elementen aus

$$\mathcal{E}(i+1, s+1, (\mathfrak{p}, \{s\}), t, \Gamma^{(g_{i+1}, i+1)})$$

zu zeigen. Wir beobachten hierzu die Wirkung auf die i–te Spalte und setzen

$$g = \left(\left(\begin{pmatrix} I_s & a^T & A \\ 0 & b & B \\ 0 & c^T & C \end{pmatrix}, 1_n \right); \tau^0 \right)$$

[2] die Vertauschbarkeit der Erzeuger ermöglicht erst den Beweis

mit $\begin{pmatrix} a^T & A \end{pmatrix} \in GF(q)^{s\times k-s}, \begin{pmatrix} b & B \\ c^T & C \end{pmatrix} \in GL_{k-s}(q), a \in GF(q)^s, b \in GF(q)$, $c \in GF(q)^{k-s-1}$, $A \in GF(q)^{s\times k-s-1}, B \in GF(q)^{k-s-1}, C \in GF(q)^{k-s-1\times k-s-1}$.
Es gilt:

$$e_s^T = \left[\left(\left(\left(\begin{pmatrix} I_s & a^T & A \\ 0 & b & B \\ 0 & c^T & C \end{pmatrix}, \varrho\right); \tau^0\right) \cdot \Gamma^{(g_{i+1},i+1)}\right](i)^T = \varrho(i) \cdot \begin{pmatrix} a & b & c \end{pmatrix}^T$$
$$\Rightarrow \varphi(i)^{-1} \cdot e_s^T = \begin{pmatrix} a & b & c \end{pmatrix}^T$$
$$\Rightarrow b = \varrho(i)^{-1}, a = 0_s, c = 0_{k-s-1}$$
$$\Rightarrow C \in GL_{k-s-1}(q)$$

Wir können das Produkt durch geeignete Elemente in $\mathcal{E}(i{+}1, s{+}1, \mathfrak{p}\cup\{\{s\}\}, t, \Gamma^{(g_{i+1},i+1)})$ ausdrücken:

$$\Rightarrow g \cdot ((I_k, \varrho); \tau^0) \cdot \underbrace{((I_k, \widetilde{\varrho}); \tau^0)}_{\in \mathcal{C}(i+1)} =$$
$$\underbrace{\left(\left(\left(\begin{pmatrix} I_{s+1} & A \\ 0 & C \end{pmatrix}, 1_n\right); \tau^0\right)\right)}_{\in \mathcal{G}(s+1)} \cdot \underbrace{\left(\left(\left(\left(\begin{pmatrix} & & 0 & & \\ & I_s & \vdots & 0 & \\ & & 0 & & \\ 0 & \dots\ 0 & \varrho(i)^{-1} & 0\ \dots\ 0 & \\ & & 0 & & \\ & 0 & \vdots & I_{k-s-1} & \\ & & 0 & & \end{pmatrix}, \varrho \cdot \widetilde{\varrho}\right); \tau^0\right)\right)\right)}_{\in \mathcal{Z}(\{s\}, \Gamma^{(g_{i+1},i+1)}))}$$

mit

$$\widetilde{\varrho}(j) := \begin{cases} \varrho(i), & j > i \wedge \exists \mu \in GF(q)^* : \Gamma^{(g_{i+1},i+1)}(j) = \mu \cdot e_s \\ 1, & \text{sonst} \end{cases}$$

Wir haben also insgesamt bewiesen:

$$\left\langle \mathcal{E}(i{+}1, s{+}1, \mathfrak{p}\cup\{\{s\}\}), t, \Gamma^{(g_{i+1},i+1)}) \right\rangle = \left((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\right)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}.$$

Da jeder Körperautomorphismus 0 und 1 fest lässt und t nach Induktionsvoraussetzung minimal ist, erhalten wir mit $(s+1, \mathfrak{p}\cup\{\{s\}\}, t)$ das Stabilisatortripel der ersten $i+1$ Spalten von $\Gamma^{(g_i,i+1)}$. □

4.1.24 Hilfssatz. *Sei $(s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)})$ ein Stabilisatortripel der ersten i Spalten von $\Gamma^{(g_i,i)}$. Desweiteren gelte:*

$$\Gamma^{(g_i,i)}(i)^T \in \left\langle \Gamma^{(g_i,i)}(0)^T, \dots, \Gamma^{(g_i,i)}(i-1)^T \right\rangle.$$

Der Algorithmus 4.1 berechnet einen $(i+1)$ – minimalen Repräsentanten $\Gamma^{(g_{i+1},i+1)}$ der Bahn $\omega(g_{i+1}\Gamma)$ auf Ebene $i+1$, sowie das Stabilisatortripel $(s^{(i+1)}, \mathfrak{p}^{(i+1)}, t^{(i+1)})$ der ersten $(i+1)$ Spalten von $\Gamma^{(g_{i+1},i+1)}$.

Algorithmus 4.1 Minimierung einer Spalte im Erzeugnis

Input: $\Gamma^{(g_i,i)}$ i–minimaler Repräsentant auf Ebene i
Input: $(s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)})$ das Stabilisatortripel der ersten i Spalten von $\Gamma^{(g_i,i)}$
Output: $\Gamma^{(g_{i+1},i+1)}$ zulässiger $(i+1)$ – minimaler Repräsentant auf Ebene i
Output: $(s^{(i+1)}, \mathfrak{p}^{(i+1)}, t^{(i+1)})$ das Stabilisatortripel der ersten $i+1$ Spalten von $\Gamma^{(g_{i+1},i+1)}$

```
Γ ← Γ^(g_i,i);
t ← t^(i);
𝔭 ← 𝔭^(i);
union ← NIL;
for j ← s^(i) − 1 to 0 do
    if Γ[j,i] ≠ 0 then
        Wähle λ ∈ l, so dass j ∈ p_λ;
        if union ≠ NIL ∧ λ ≠ union then
            // Bringe Eintrag auf 1 durch gekoppelte Zeilen–/Spaltenmultiplikation
            Γ ← z_λ(Γ[j,i]^{-1}) · Γ;
            p_union ← p_union ∪ p_λ;
            p_λ ← ∅;
        end if
        if λ = union then
            Wähle Potenz m ∈ {1,... r/t} so dass
              τ^{tm}(Γ[j,i]) minimal in der Menge {τ^{ta}(Γ[j,i]) | a ∈ {1,... r/t}};
            Γ ← τ^{tm}(Γ);
            Wähle minimales t' ∈ {1,... r/t} so dass τ^{tt'}(Γ[j,i]) = Γ[j,i];
            t ← tt';
        end if
        if union = NIL then
            union ← λ;
            // Bringe Eintrag auf 1 durch Multiplikation der Spalte
            Γ(i) ← Γ[j,i]^{-1} · Γ(i);
        end if
    end if
end for
Entferne aus 𝔭 alle p_λ mit p_λ = ∅;
return (Γ, (s^(i), 𝔭, t));
```

Beweis. Es sei $\mathfrak{p}^{(i)} = \{p_0, \ldots, p_{l^{(i)}-1}\}$. Da die ersten i Spalten nach Induktionsvoraussetzung die ersten $s^{(i)}$ Einheitsvektoren beinhalten und der Rang der Matrix $\Pi_i(\Gamma^{(g_i,i)}) = s^{(i)}$

ist, sind die Einträge der Spalte i in den Zeilen $\{s^{(i)}, \ldots, k-1\}$ gleich Null. Jede Anwendung eines Gruppenelements $g \in \mathcal{G}(s^{(i)})$ verändert somit nicht die Spalte mit Index i. Genauso wirkt eine gekoppelte Zeilen-/Spaltenmultiplikation zu $\mathcal{Z}(p, \Gamma^{(g_i,i)})$ mit $p \cap supp(\Gamma^{(g_i,i)}(i)) = \emptyset$ nicht auf diese Spalte, da ausschließlich Nulleinträge multipliziert werden. Weiterhin lässt eine beliebige Multiplikation $c \in \mathcal{C}(i+1)$ offensichtlich die Matrix $\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})$ fest.
Durch die Definition des Algorithmus, lässt der Körperautomorphismus $\tau^{t^{(i+1)}}$ die Matrix $\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})$ genauso fix, wie eine gekoppelte Zeilen-/Spaltenmultiplikation $z \in \mathcal{Z}(p_{union}, \Gamma^{(g_{i+1},i+1)})$. Somit ist

$$\mathcal{E}(i+1, s^{(i+1)}, \mathfrak{p}^{(i+1)}, t^{(i+)}, \Gamma^{(g_{i+1},i+1)}) \subseteq \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}$$

und es bleibt die Erzeugereigenschaft zu zeigen. Es sei hierzu $c \in \mathcal{C}(i) \backslash \mathcal{C}(i+1)$, $a \in \mathcal{A}(t^{(i)})$ und $z_\lambda \in \mathcal{Z}(p_\lambda, \Gamma^{(g_i,i)}), \lambda \in \Lambda := \{\lambda' \in l^{(i)} \mid p_{\lambda'} \cap supp(\Gamma^{(g_i,i)}(i)) \neq \emptyset\}$. Wir definieren zu $\lambda \in \Lambda$ die Zeilen

$$j_\lambda := \max\big\{j' \in p_\lambda \cap supp\big(\Gamma^{(g_{i+1},i+1)}\big)\big\}.$$

Die Einträge der Spalte $\Gamma^{(g_{i+1},i+1)}(i)^T$ sind an diesen Positionen wegen der Arbeitsweise des Algorithmus gleich Eins:

$$\forall \lambda \in \Lambda : \Gamma^{(g_{i+1},i+1)}_{j_\lambda,i} = 1.$$

Nach obigen Aussagen und Hilfssatz 4.1.15 reicht es für ein Produkt

$$c \cdot \prod_{\lambda \in \Lambda} z_\lambda \cdot a \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}$$

die Erzeugbarkeit zu beweisen. Wir beobachten hierzu die Wirkung auf die Spalte $\Gamma^{(g_{i+1},i+1)}(i)^T$:

1.Fall, $|\Lambda| = 1$**:** Es sei $\Lambda = \{\lambda_0\}$. Offensichtlich ist somit auch die Multiplikation mit z_{λ_0} bereits ein Element von $\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}$. Da im Algorithmus keine Zeilenindexmengen verschmolzen werden, ist aber $\mathfrak{p}^{(i)} = \mathfrak{p}^{(i+1)}$. Damit liegt auch $z_{\lambda_0} \in \mathcal{E}(i+1, s^{(i+1)}, \mathfrak{p}^{(i+1)}, t^{(i+)}, \Gamma^{(g_{i+1},i+1)})$ und somit reicht es für das Produkt

$$c \cdot a = z_{\lambda_0}^{-1} \cdot c \cdot z_{\lambda_0} \cdot a \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}$$

die Erzeugbarkeit zu zeigen. Da jeder Körperautomorphismus das Einselement des Körpers fix lässt, muss die Multiplikation mit c trivial sein. Somit gilt aber bereits $a \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}$. Nach der Definition des Algorithmus ist $\tau^{t^{(i+1)}}$ aber der minimale Erzeuger der Körperautomorphismen, welche die Matrix $\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})$ fest lassen. Also gilt $a \in \mathcal{A}(t^{(i+1)})$ und damit die Behauptung.

2.Fall, $|\Lambda| > 1$**:** Die Multiplikation mit c bewirke eine Multiplikation der Spalte $\Gamma^{(g_{i+1},i+1)}(i)^T$ mit $\mu \in GF(q)^*$. Da außerdem $supp\big(\Gamma^{(g_{i+1},i+1)}\big) \not\subseteq p_\lambda, \forall \lambda \in \Lambda$ wird bei der Multiplikation mit z_λ die Spalte $\Gamma^{(g_{i+1},i+1)}(i)^T$ mit Eins multipliziert. Für die Zeile j_λ sei durch die Multiplikation mit z_λ eine Multiplikation der Einträge mit μ_λ gegeben. Es gilt also für jedes $\lambda \in \Lambda$:

$$\begin{aligned} 1 = \Gamma^{(g_{i+1},i+1)}_{j_\lambda,i} &= \mu \cdot \mu_\lambda \cdot \alpha\big(\Gamma^{(g_{i+1},i+1)}_{j_\lambda,i}\big) = \mu \cdot \mu_\lambda \\ \Rightarrow \mu^{-1} &= \mu_\lambda \end{aligned}$$

Damit ist aber

$$c \cdot \prod_{\lambda\in\Lambda} z_\lambda = z^{(i+1)}_{union} \cdot c^{(i+1)}$$

für $z^{(i+1)}_{union} \in \mathcal{Z}(p_{union}, \Gamma^{(g_{i+1},i+1)})$ und $c^{(i+1)} \in \mathcal{C}(i+1)$. Analog zu oben gilt damit aber auch $a \in \mathcal{A}(t^{(i+1)})$ und damit der Beweis der Behauptung.

Die $(i+1)$ – Minimalität ergibt sich analog. Wir geben uns ein Gruppenelement in der Zerlegung wie oben vor:

$$c \cdot \prod_{\lambda\in\Lambda} z_\lambda \cdot a \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_i,i)})}$$

so dass die Matrix

$$\big(c \cdot \prod_{\lambda\in\Lambda} z_\lambda \cdot a\big)\Gamma^{(g_i,i)}$$

$(i+1)$–minimal ist und zeigen, dass die Spalten mit Index i gleich sind:

$$\Big(\big(c \cdot \prod_{\lambda\in\Lambda} z_\lambda \cdot a\big)\Gamma^{(g_i,i)}\Big)(i)^T = \Gamma^{(g_{i+1},i+1)}(i)^T.$$

Eine erste Beobachtung ergibt für die Spalten i eines jeden $(i+1)$–minimalen Repräsentanten, dass die Einträge in den Zeilen $j_\lambda, \lambda \in \Lambda$ gleich Eins sein müssen. Wegen $\tau(1) = 1$ folgt aber somit aus obigem Beweis, dass das Produkt $c \cdot \prod_{\lambda\in\Lambda} z_\lambda$ nur aus der Gruppe $\big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})}$ gewählt werden kann. Die elementweise Minimierung durch a wird aber gerade durch den Algorithmus abgedeckt. Somit ist die Matrix $\Gamma^{(g_{i+1},i+1)}$ ein $(i+1)$ – minimaler Repräsentant der Bahn $\omega(g_i\Gamma)$. □

4.1.25 Bemerkung. Ist $i = n$, so können wir aus dem Stabilisatortripel $(s, \mathfrak{p}, t)$ des n–minimalen Repräsentanten $\Gamma^{(n)}$ neben dem Erzeugendensystem noch die folgenden Informationen ablesen:

- Gilt $|\mathfrak{p}| > 1$, so ist der Code zerlegbar,
- ist $|\mathfrak{p}| = 1$, so ist das einzige Element $p_0 = \{0, \ldots, k-1\}$ und der Code unzerlegbar,
- falls $t \neq r$, so führt die Untergruppe $\mathcal{A}(t)$ der Körperautomorphismen den Code auf sich selbst über.

Beweis. Da Γ mit vollem Zeilenrang k vorausgesetzt wurde, ist $s = k$. Somit ist $\mathfrak{p}$ eine Partition der Zeilenindexmenge. Desweiteren sind alle Einheitsvektoren Spalten von $\Gamma^{(n)}$ und somit erreichen wir durch eine geeignete Permutation $\sigma \in S_n$ der Spalten eine systematische Generatromatrix:

$$\sigma\Gamma^{(n)} = \begin{pmatrix} I_k & A \end{pmatrix}.$$

Wir wenden den Test 6.2.13 aus [BBF+06] mit folgender Aussage an:
$C(\Gamma^{(n)})$ *ist unzerlegbar genau dann, wenn es eine Folge* $a_{j,j}, a_{l,m}, \ldots$ *von Nichtnulleinträgen der Matrix A gibt, so dass jedes Element der Folge in der gleichen Spalte oder Zeile liegt wie sein Vorgänger und jede Zeile mindestens einmal repräsentiert wird.*
Ist $|\mathfrak{p}| = 1$, so können wir nach Algorithmus 4.1 die Verschmelzungsvorschriften der p_i zu einer solchen Folge zusammenfassen.
Ist jedoch $|\mathfrak{p}| > 1$, so gilt für zwei disjunkte Mengen $p_i, p_j \in \mathfrak{p}$ und jede Spalte in $A(l)^T$:

$$\mathrm{supp}(A(l)^T) \cap p_i \neq \emptyset \Rightarrow \mathrm{supp}(A(l)^T) \subseteq p_i \Rightarrow \mathrm{supp}(A(l)^T) \cap p_j = \emptyset.$$

Damit gibt es aber keine Folge obiger Gestalt, die die Zeilen in p_i mit den restlichen verknüpfen kann. Der Code $C(\Gamma^{(n)})$ ist also zerlegbar und somit auch C. Ein sukzessives Anwenden des Tests ergibt sogar:

$$|\mathfrak{p}| = l \Rightarrow C \simeq C_0 \dot{+} \ldots \dot{+} C_{l-1}$$

mit unzerlegbaren Codes $C_i, i \in l$. □

4.1.26 Bemerkung. Setzen wir nun voraus, dass die Multiplikation, Addition, die Berechnung der Inversen sowie des Bildes des Frobenius Automorphismus in konstanter Zeit unabhängig von q geschehe, etwa durch Nachschlag in entsprechenden Tabellenwerken. Bei Übergang zu einem Knoten i ist eine Spalte unter dem Stabilisator der ersten i Spalten zu minimieren. Der Aufwand zur Berechnung des Gaußschritts erfolgt in diesem Fall unabhängig von q, bei der Berechnung gemäß Algorithmus 4.1 erfolgen die Multiplikationen wiederum unabhängig von q, jedoch sind die Bahnen eines Elements zu Durchlaufen. Wir haben also eine Abhängigkeit von r, vielmehr von der Menge der Teiler $\{t \text{ teilt } r\}$ von r, nicht jedoch der Größenordnung von q. Bei der Laufzeitanalyse im Anhang, ergibt sich aber trotzdem eine Abhängigkeit von q. Dies ist auf die Güte der gewählten Homomorphismen zurückzuführen, die mit wachsender Körperordnung abnimmt, siehe hierzu Anhang A.1.

4.2. Verwaltung der Stabilisatoren

4.2.1 Definition. Eine Folge $[\alpha_0, \ldots, \alpha_{m-1}]$ natürlicher Zahlen α_j mit $\sum_{j \in m} \alpha_j = n$ heißt Zahlpartition von n.
Sei $[\alpha_0, \ldots, \alpha_{m-1}]$ eine Zahlpartition von n, dann heißt die Gruppe

$$S_{[\alpha_0,\ldots,\alpha_{m-1}]} := S_{\{0,\ldots,\alpha_0-1\}} \oplus S_{\{\alpha_0,\ldots,\alpha_0+\alpha_1-1\}} \oplus \ldots \oplus S_{\{\alpha_0+\ldots+\alpha_{m-2},\ldots,n-1\}}$$

die kanonische Younguntergruppe zur Zahlpartition $[\alpha_0, \ldots, \alpha_{m-1}]$.

4.2.2 Bemerkung. Die kanonischen Younguntergruppen sind genau die Stabilisatoren der geordneten Mengenpartitionen

$$\{0, \ldots, \alpha_0 - 1\}, \{\alpha_0, \ldots, \alpha_0 + \alpha_1 - 1\}, \ldots, \{\alpha_0 + \alpha_1 + \ldots + \alpha_{m-2}, \ldots, n-1\}$$

von n unter elementweiser Operation der S_n.

4.2.3 Definition. Ein G–Homomorphismus $f : X \to Y$ heißt G–Invariante, falls die Operation von G auf Y trivial ist. Dies ist äquivalent zu der Forderung

$$\forall g \in G : f(gx) = f(x).$$

4.2.4 Hilfssatz. *Es sei Y^n eine beliebige Menge. Durch die Operation der symmetrischen Gruppe auf n ist auch eine Operation von S_n auf Y^n gegeben. Weiter sei $H := S_{[\alpha_0,\ldots,\alpha_{m-1}]} \leq S_n$ eine kanonische Younguntergruppe. Auf der Menge Y sei eine Totalordnung $(Y, \leq)$ gegeben. Die Vektoren in Y^n seien lexikographisch bzw. revers lexikographisch angeordnet.*
Der Stabilisator H_y eines $y \in Can_{min}(H)$ ist wieder eine kanonische Younguntergruppe.

Beweis. Innerhalb eines Blocks $\{\beta, \ldots, \gamma\} := \left\{\sum_{i<l} \alpha_i, \ldots, \left(\sum_{i \leq l} \alpha_i\right) - 1\right\}, l \in m$ der geordneten Mengenpartition zu $[\alpha_0, \ldots, \alpha_{m-1}]$ müssen die Einträge $(y_\beta, y_{\beta+1}, \ldots, y_\gamma)$ lexikographisch aufsteigend bzw. absteigend angeordnet sein. Der Stabilisators H_y besteht also aus den beliebigen Permutationen der gleichen Einträge eines Blocks. Diese Koordinaten bilden aber gerade eine geordnete Mengenpartition, der Stabilisator hierzu ist wieder eine kanonische Younguntergruppe. □

Im Algorithmus werden wir als $S_n^{(i)}$–Homomorphismen entweder Homomorphismen mit Bildbereichen von dieser Form wählen oder $S_n^{(i)}$–Invarianten vorschreiben. Da auch die Schnitte von kanonischen Younguntergruppe mit den Untergruppe $S_n^{(i)}$ wieder kanonische Younguntergruppe ergeben, sind die vorkommenden Untergruppen $H_{hgx}^{(i,j)}$ stets kanonische Younguntergruppen. Wir können diese also durch geordnete Mengenpartitionen von n verwalten.

Die Gruppe $H_{hgx}^{(i,j_i)}$ ist insbesondere eine Untergruppe von

$$S_n^{(i)} = S_{\{0\}} \oplus \ldots \oplus S_{\{i-1\}} \oplus S_{\{i,\ldots,n-1\}},$$

also muss die darstellende geordnete Mengenpartition zu $H_{hgx}^{(i,j_i)}$ von der folgenden Form sein, mit geeignetem $\nu \leq n$:

$$\{0\}, \{1\}, \ldots, \{i-1\}, \{i, i+1, \ldots, \nu-1\}, \ldots.$$

Wir können die benötigte Transversale $\overline{T}^{(i)}$ von $H_{hgx}^{(i+1,0)} \backslash H_{hgx}^{(i,j_i)}$ wie folgt wählen:

$$\overline{T}^{(i)} := \{\mathrm{id}, (i, i+1), (i, i+2), \ldots, (i, \nu-1)\}$$

oder

$$\overline{T}^{(i)} := \{\mathrm{id}, (i, i+1), (i, i+1, i+2), \ldots, (i, i+1, \ldots, \nu-1)\}$$

4.2.5 Bemerkung. Da wir in der Implementierung die Berechnungen ausgehend von den Brüdern starten, bevorzugen wir die zweite Darstellung. Hier erreichen wir stets den nächsten Knoten über eine Vertauschung von zwei Spalten. Für den ersten Sohn müssen wir ohnehin die Ausgabe von $can_{\overline{Y}_i}$ berücksichtigen.
Wir werden während des Algorithmus stets die i'–minimalen Repräsentanten $\Gamma^{(g_{i'}, i')}$ für jeden Vorgänger $g_{i'}$ von g_i auf Ebene $i' \leq i$ abspeichern. Findet nun im Backtrackdurchlauf ein Rücksprung auf Ebene $\kappa \leq i$ statt und ist dort der Index der eingehenden Kante j, so erhalten wir durch

$$\widetilde{\Gamma}^{(g_\kappa, \kappa)} := (\kappa - 1, \kappa - 1 + j)\Gamma^{(g_\kappa, \kappa)}$$

ein Element der Bahn $\omega(\overline{T}^{(\kappa-1)}[j] \cdot g_{\kappa-1}\Gamma)$. Durch die Minimierung der Spalte κ von $\widetilde{\Gamma}^{(g_\kappa, \kappa)}$ gemäß Abschnitt 4.1 erhalten wir einen κ–minimalen Repräsentanten $\Gamma^{(\overline{T}^{(\kappa-1)}[j] \cdot g_{\kappa-1}, \kappa)}$ der inneren Bahn $\omega(\overline{T}^{(\kappa-1)}[j] \cdot g_{\kappa-1}\Gamma)$.

4.3. Homomorphieprinzip

4.3.1. Die benutzten Homomorphismen auf Ebene 0

Wir werden auf Ebene 0 zwei Homomorphismen zum Einsatz bringen. Zunächst können wir die Spalten nach Nullspalten und Nichtnullspalten sortieren:

4.3.1 Satz. *Die Abbildung*

$$f_{0,0} : \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q) \Big) \backslash\backslash GF(q)^{k \times n, k} \to 2^n$$
$$\omega(\Gamma) \mapsto f_{0,0}(\omega(\Gamma))$$

mit $f_{0,0}(\omega(\Gamma))(j) := \begin{cases} 0, \Gamma(j) = 0_k \\ 1, \text{sonst} \end{cases}$ *ist wohldefiniert und ein* S_n*–Homomorphismus.*

Beweis. Ist $\Gamma' \in \omega(\Gamma)$ ein weiterer Repräsentant der Bahn, so existiert ein $((A,\varphi);\alpha) \in (GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)$ mit $((A,\varphi);\alpha)\Gamma = \Gamma'$. Für eine beliebige Spalte $j \in n$ gilt:

$$\Gamma'(j)^T = \big(((A,\varphi);\alpha)\Gamma\big)(j)^T = \varphi(j)A\alpha(\Gamma(j)^T)$$

Der Körperautomorphismus α, die Multiplikation mit der invertierbaren Matrix A und die Multiplikation mit Einheiten $\varphi(j)$ bilden jeweils genau die Nullspalten auf Nullspalten ab. Somit ist die Zuordnung unabhängig von der Wahl des Repräsentanten Γ.

$$\Rightarrow \forall \Gamma' \in \omega(\Gamma) : f_{0,0}(\omega(\Gamma')) = f_{0,0}(\omega(\Gamma)).$$

Weiterhin gilt für ein beliebiges $\pi \in S_n$:

$$\begin{aligned} f_{0,0}(\pi\omega(\Gamma))(j) =& f_{0,0}(\omega(\pi\Gamma))(j) = \begin{cases} 0, (\pi\Gamma)(j) = 0_k \\ 1, \text{sonst} \end{cases} \\ =& \begin{cases} 0, \Gamma(\pi^{-1}(j)) = 0_k \\ 1, \text{sonst} \end{cases} \quad = \big(\pi f_{0,0}(\omega(\Gamma))\big)(j). \end{aligned}$$

Also ist $f_{0,0}$ wie behauptet ein S_n–Homomorphismus. □

Wir wählen für den Bildbereich 2^n die revers lexikographische Anordnung der Vektoren bezüglich der natürlichen Anordnung $0 < 1$ auf der Menge $\{0,1\}$. Die Anwendung des Homomorphismus $f_{0,0}$ im Backtrackverfahren ergibt somit eine Permutation $h_{0,0}$ und eine Partition

$$\{0,\ldots,n'-1\},\{n',\ldots,n-1\}$$

von n, wobei genau die ersten n' Spalten von $h_{0,0}\Gamma$ keine Nullspalten der Generatormatrix sind. Im weiteren Verlauf werden nur noch Permutationen aus $S_{[n',n-n']}$ zugelassen. Die kanonischen Generatormatrizen müssen also die Sortierung nach Nichtnullspalten und Nullspalten einhalten. Aufgrund dieser Beobachtungen können wir auch direkt mit der Kanonisierung von $\Pi_{n'}(\Gamma)$ in $(GL_k(q) \times GF(q)^{*n'}) \rtimes Gal(q)$ starten und die Ausgabe $\widetilde{\Gamma}$ durch Anfügen der verblieben $n-n'$ Nullspalten zu einer Lösung des Ausgangsproblem ergänzen.

4.3.2 Hilfssatz. *Es sei $\Gamma \in GF(q)^{k\times n,k}$ eine beliebige Generatormatrix mit $n-n'$ Nullspalten. Weiterhin sei $h_{0,0} \in S_n$ eine Sortierung der Spalten, so dass die Nichtnullspalten zu Beginn stehen. Zu $\Pi_{n'}(h_{0,0}\Gamma)$ sei die Automorphismengruppe*

$$Aut\big((\Pi_{n'}(h_{0,0}\Gamma)\big) \leq \big(GL_k(q) \times GF(q)^{*n'}\big) \rtimes (Gal(q) \times S_{n'})$$

bekannt. Die Automorphismengruppe $Aut\big(h_{0,0}\Gamma\big)$ wird erzeugt von:

$$\begin{aligned} &\big\{((A,\varphi);(\alpha,\pi)) \in \big(GL_k(q) \times GF(q)^{*n}\big) \rtimes (Gal(q) \times S_n)\big| \\ &\quad ((A,\varphi\downarrow_{n'});(\alpha,\pi)) \in Aut\big((\Pi_{n'}(h_{0,0})\big)\big\} \\ \cup\ &\big\{((I_k,1_n);(\tau^0,\pi)) \in \big(GL_k(q) \times GF(q)^{*n}\big) \rtimes (Gal(q) \times S_n)\big|\pi \in S_{\{n',\ldots,n-1\}}\big\}. \end{aligned}$$

Beweis. Trivial. □

4.3.3 Folgerung. *Wir starten das Backtrackverfahren nicht für* Γ*, sondern für die Matrix* $\Pi_{n'}(h_{0,0}\Gamma)$ *und ergänzen die Lösung zu einer Gesamtlösung durch Anfügen von* $n - n'$ *Nullspalten.*

Im Folgenden gehen wir also stets vom Vorliegen einer nicht redundanten Generatormatrix aus. Insofern, lassen sich auch die oben eingeführten Stabilisatortripel zur Verwaltung anwenden.

Neben dem Aussortieren der Nullspalten können wir noch einen Schritt auf Ebene 0 durchführen, der die Länge n unter Umständen weiter reduzieren kann.

4.3.4 Satz. *Die Abbildung*

$$\begin{aligned}\widetilde{f}_{0,1} :\Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big)\backslash\backslash GF(q)^{k\times n,k} &\to \mathbb{N}^n\\ \omega(\Gamma) &\mapsto \widetilde{f}_{0,1}(\omega(\Gamma))\end{aligned}$$

mit $\widetilde{f}_{0,1}(\omega(\Gamma))(j) := \big|\{l \in n \mid \exists \mu \in GF(q)^* : \mu\Gamma(l) = \Gamma(j)\}\big|$ *ist wohldefiniert und ein* S_n*–Homomorphismus.*

Beweis. Es seien $j, l \in n$ mit $\Gamma(j) = \mu\Gamma(l)$ für ein $\mu \in GF(q)^*$. Weiter sei $((A, \varphi); \alpha) \in (GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)$ beliebig:

$$\begin{aligned}(((A, \varphi); \alpha)\Gamma)(j)^T =&\varphi(j)A\Gamma(j)^T = \varphi(j)A(\mu\Gamma(l)^T) = \mu\varphi(j)\varphi(l)^{-1}\varphi(l)A\Gamma(l)^T\\ =&\underbrace{\mu\varphi(j)\varphi(l)^{-1}}_{\in GF(q)^*}(((A, \varphi); \alpha)\Gamma)(l)^T\end{aligned}$$

Die Abbildung ist somit unabhängig von dem gewählten Repräsentanten Γ. Es gilt weiter für jedes $\pi \in S_n, j \in n$:

$$\begin{aligned}\widetilde{f}_{0,1}(\pi\omega(\Gamma))(j) =&\widetilde{f}_{0,1}(\omega(\pi\Gamma))(j) = \big|\{l \in n \mid \exists \mu \in GF(q)^* : \mu \cdot (\pi\Gamma)(l) = (\pi\Gamma)(j)\}\big|\\ =&\big|\{l \in n \mid \exists \mu \in GF(q)^* : \mu\Gamma(l) = \Gamma(\pi^{-1}(j))\}\big|\\ =&\widetilde{f}_{0,1}(\omega(\Gamma))(\pi^{-1}(j)) = (\pi\widetilde{f}_{0,1}(\omega(\Gamma)))(j)\end{aligned}$$

□

Wir wenden den Homomorphismus aber nicht direkt an, sondern gewinnen hieraus zunächst:

4.3.5 Satz. *Die Abbildung*

$$\begin{aligned}f_{0,1} :\Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big)\backslash\backslash GF(q)^{k\times n,k} &\to \mathbb{N}^n\\ \omega(\Gamma) &\mapsto f_{0,1}(\omega(\Gamma))\end{aligned}$$

mit

$$f_{0,1}(\omega(\Gamma))(j) := \frac{\left|\{l \in n \mid \widetilde{f}_{0,1}(\omega(\Gamma))(l) = \widetilde{f}_{0,1}(\omega(\Gamma))(j)\}\right|}{\widetilde{f}_{0,1}(\omega(\Gamma))(j)}$$

ist wohldefiniert und ein S_n–Homomorphismus. Wir wählen als Anordnung auf $\mathbb{N}^n$ die lexikographische Ordnung.

Beweis. Folgt aus den Eigenschaften von $\widetilde{f}_{0,1}$. □

Erst als dritten Homomorphismus wenden wir $f_{0,2} = \widetilde{f}_{0,1}$ selbst an. Auch hier verwenden wir im Bildbereich $\mathbb{N}^n$ die lexikographische Ordnung.

Die Begründung für diese Wahl erhalten wir, wenn wir die beiden Homomorphismen als Einheit sehen. Aufgrund von $\widetilde{f}_{0,1}$ wissen wir, dass nur Spalten aufeinander abgebildet werden dürfen, die die gleiche Anzahl von linear abhängigen Spalten in Γ aufweisen. Insbesondere führt aber die Vertauschung von zwei linear abhängigen Spalten mit entsprechenden Multiplikationen der betroffenen Spalten zu einem Automorphismus des Codes. Wir müssen also diese Untergruppe nicht erst durch das Backtracking berechnen. Wir bilden aus der ursprünglichen Generatormatrix Γ die Generatormatrix $\widetilde{\Gamma} \in GF(q)^{k \times n'}$ durch Auswahl von je einem Repräsentanten der eindimensionalen Unterräume $\{\langle \Gamma(j)^T \rangle \mid j \in n\}$. Als operierende Gruppe lassen wir aber nur noch Permutationen zu, die die Vielfachheiten erhalten, also eine kanonische Younguntergruppe $S_{[\alpha_0,\ldots,\alpha_{m-1}]}$. Durch das Vorschreiben des Homomorphismus $f_{0,1}$ erreichen wir, dass die Zahlpartition $[\alpha_0, \ldots, \alpha_{m-1}]$ aufsteigend sortiert ist bzw. ein kleinster Block zuerst betrachtet wird. Dadurch erhält man unterhalb der Wurzel zunächst weniger Söhne. Dies ist insbesondere deshalb sehr nützlich, da die Homomorphismen in Abschnitt 4.3.2 erst ab $i \geq 2$ greifen und mit wachsendem i – vielmehr mit wachsendem Rang s der i-ten Projektion – an Bedeutung gewinnen. Für die dort aufgeführten Homomorphismen wurde auf die Sortierung nach Blockgröße verzichtet, da nicht abzusehen ist, wie sich die Partition im weiteren Verlauf des Algorithmus entwickelt und sich somit ein zu großer Aufwand ergäbe.

Aus dem kanonischen Repräsentanten zu $\widetilde{\Gamma}$ erhalten wir rückwirkend wieder einen kanonischen Repräsentanten für Γ durch entsprechendes Vervielfachen der Spaltenanzahlen und Beibehaltung der Reihenfolge. Für das kanonisierende Element und die Automorphismengruppe muss die Ausgabe des Kanonisierers ebenfalls nur entsprechend auf die Ausgangsmatrix zurückgeführt werden. Die nachfolgenden Hilfssätze formulieren ein systematisches Vorgehen.

4.3.6 Definition. Wir nennen zu einem Vektor $v \in GF(q)^k$ den Index

$$lc(v) := \max\{l \in k \mid v_l \neq 0\}$$

den Leitkoeffizienten von v. Ein Vektor $v \in GF(q)^k$ heißt normiert, falls $lc(v) = 1$.

4.3.7 Hilfssatz. *Es sei $\widetilde{\Gamma} \in GF(q)^{k\times n'}$ die aus Γ durch obige Auswahl von Spalten gewonnene Matrix. Wir definieren zu $j \in n'$ die ursprünglichen Koordinaten*

$$\mathcal{C}_j := \{l \in n \mid \exists \mu \in GF(q)^* : \Gamma(l) = \mu\widetilde{\Gamma}(j)\}.$$

Ohne Beschränkung der Allgemeinheit sind die Spalten von Γ und $\widetilde{\Gamma}$ normiert und die linear abhängigen Spalten in der Reihenfolge von $\widetilde{\Gamma}$ sortiert, dh. es gelte für beliebige $j', m' \in n', j' < m'$:

$$j \in \mathcal{C}_{j'}, m \in \mathcal{C}_{m'} \Rightarrow j < m$$

Schließlich sei noch $\pi \in S_{n'}$ eine Permutation, die die Vielfachheiten erhält: $\pi(j') = m' \Rightarrow |\mathcal{C}_{j'}| = |\mathcal{C}_{m'}|$ und $((A,\varphi);(\alpha,\pi)) \in Aut(\widetilde{\Gamma})$. Dann ist durch

$$((A,\varphi);(\alpha,\pi))^\Gamma := ((A,\rho);(\alpha,\sigma))$$

mit

$$\rho \in GF(q)^{*n} \text{ sd. } \forall j' \in n', \forall j \in \mathcal{C}_{j'} : \rho(j) := \varphi(j')$$
$$\pi \in S_n \text{ sd. } \forall j' \in n', \forall i \in |\mathcal{C}_{j'}| : \sigma(i + \min \mathcal{C}_{j'}) := i + \min \mathcal{C}_{\pi(j')}$$

ein Automorphismus von Γ definiert.

Beweis. Es gilt:

$$\forall j' \in n' : \sigma(\mathcal{C}_{j'}) = \mathcal{C}_{\pi(j')}$$

und damit für eine beliebige Spalte $j \in \mathcal{C}_{j'}$:

$$\begin{aligned}\Gamma(j)^T &= \big(((A,\rho);(\alpha,\sigma))\Gamma\big)(j)^T = \rho(j)\cdot A\alpha\big(\Gamma(\sigma^{-1}(j))^T\big)\\ &= \varphi(j')A\alpha\big(\widetilde{\Gamma}(\pi^{-1}(j'))^T\big) = \widetilde{\Gamma}(j')^T = \Gamma(j)^T\end{aligned}$$

□

4.3.8 Folgerung. *Seien die Voraussetzungen wie oben, dann erhalten wir ein Erzeugendensystem von $Aut(\Gamma)$ durch*

$$\begin{aligned}E :=&\{((A,\varphi);(\alpha,\pi))^\Gamma \mid ((A,\varphi);\alpha,\pi) \in Aut(\widetilde{\Gamma}) \wedge \forall j' \in n' : |C_{j'}| = |C_{\pi(j')}|\}\\ &\cup \bigcup_{j\in n'} \{((I_k,1_n);(\tau^0,\pi)) \mid \pi \in S_{\mathcal{C}_j}\}.\end{aligned}$$

Beweis. Für die Menge E gilt sicherlich $E \subseteq Aut(\Gamma)$. Wir müssen zeigen, dass sie auch die Automorphismengruppe erzeugt. Es sei dazu $((B,\gamma);(\beta,\psi)) \in Aut(\Gamma)$ beliebig. Wir sortieren zunächst die Spalten in den Mengen $\psi(\mathcal{C}_{j'}), j' \in n'$ durch eine Permutation $\pi_{j'} \in S_{\psi(\mathcal{C}_{j'})}$ nach der Anordnung in der Menge $\mathcal{C}_{j'}$. Der Automorphismus

$$\prod_{j' \in n'} ((I_k, 1_n); (\tau^0, \pi_{j'})) \cdot ((B,\gamma);(\beta,\psi))$$

ergibt durch Übertragung auf $\widetilde{\Gamma}$ einen Automorphismus $((A,\varphi);(\alpha,\pi)) \in Aut(\widetilde{\Gamma})$. Durch die Sortierung der Spalten werden außerdem die einzelnen Koordinaten entsprechend ihrer Anordnung in der Menge abgebildet. Somit ist

$$\begin{aligned} &\prod_{j' \in n'} ((I_k, 1_n); (\tau^0, \pi_{j'})) \cdot ((B,\gamma); \beta, \psi) = ((A,\varphi);(\alpha,\pi))^{\Gamma} \in E \\ \Rightarrow &((B,\gamma);(\beta,\psi)) = \prod_{j' \in n'} ((I_k, 1_n); (\tau^0, \pi_{j'}^{-1}))((A,\varphi);(\alpha,\pi))^{\Gamma} \in \langle E \rangle \end{aligned}$$

□

4.3.9 Folgerung. *Sind die Voraussetzungen wie oben jedoch $((A,\varphi);(\alpha,\pi))$ ein kanonisierendes Element zu $\widetilde{\Gamma}$, so erhalten wir durch die analoge Definition von ρ und σ ein kanonisierendes Element für Γ.*

Beweis. Analog zu oben. □

4.3.10 Bemerkung. Es genügt also einen Kanonisierer für projektive Generatormatrizen zur Verfügung zu stellen und entsprechende Einschränkungen auf kanonische Younguntergruppe vorzuschreiben.

4.3.11 Bemerkung. Durch den Homomorphismus $f_{0,2}$ erhalten wir eine Beschränkung der Knoten des Backtrackbaums. Durch Verschmelzen von Spalten, die lineare Vielfache voneinander sind, kann die Matrix Γ' höchstens so viele Spalten besitzen, wie es eindimensionale Unterräume in $GF(q)^k$ gibt. Es gilt also:

$$n' \leq \min\left\{\frac{q^k - 1}{q-1}, n\right\}.$$

4.3.12 Beispiel. Wir wählen zwei Repräsentanten von verschiedenen[3] semilinearen Isometrieklassen unzerlegbarer $(7,2,4,2)$ – Codes :

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

[3] nach [BBF+06]

Würden wir nur den Homomorphismus $f_{0,2}$ anwenden, so erhielten wir die folgen Matrizen:

$$\overline{\Gamma}_1 := \left(\begin{array}{c|cc} 0 & 1 & 1 \\ 1 & 1 & 0 \end{array}\right) \text{ und } \overline{\Gamma}_2 := \left(\begin{array}{cc|c} 1 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right)$$

Mit Anwendung von $f_{0,1}$ und $f_{0,2}$ jedoch

$$\widetilde{\Gamma}_1 := \overline{\Gamma}_1 \text{ und } \widetilde{\Gamma}_2 := \left(\begin{array}{c|cc} 0 & 1 & 1 \\ 1 & 1 & 0 \end{array}\right)$$

Mit den senkrechten Strichen in den Matrizen seien jeweils die Blöcke der Mengenpartitionen von n' gekennzeichnet, welche die kanonische Younguntergruppen definieren. Wir führen das Beispiel fort und berechnen einen minimalen Repräsentanten, sowie die Automorphismengruppe. Da wir im binären Fall keine Multiplikation der Spalten und keinen Körperautomorphismus berücksichtigen müssen, bleibt eine Minimierung der Matrix durch Gaußschritte. Wir erhalten als minimalen Repräsentanten

$$\widetilde{\Gamma}^{min} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \left(\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}; \mathrm{id}\right) \cdot \widetilde{\Gamma}_1$$

sowie

$$Aut(\widetilde{\Gamma}_1) \cap S_{[1,2]} = \left\{ (I_2; \mathrm{id}), \left(\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}; (12)\right) \right\}$$

Für die Rücktransformation des Ergebnisses mit obigen Sätzen müssen wir noch für eine passende Sortierung der Spalten in Γ_1 sorgen, diese erhalten wir zum Beispiel durch einen zyklischen Shift (0123456):

$$(0123456)\Gamma_1 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$
$$\Gamma_1^{min} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

sowie

$$\begin{aligned} Aut((0123456)\Gamma_1) = \Big\langle & \left\{ \left(\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}; (14)(25)(36)\right) \right\} \cup \{(I_2; \pi_1) \mid \pi_1 \in S_{\{1,2,3\}}\} \\ & \cup \{(I_2; \pi_2) \mid \pi_2 \in S_{\{4,5,6\}}\} \Big\rangle \\ \Rightarrow Aut(\Gamma_1) = & \big(I_2; (0654321)\big) \cdot Aut\big((0123456)\Gamma_1\big) \cdot \big(I_2; (0123456)\big) \\ \Rightarrow |Aut(\Gamma_1)| = & \, 2 \cdot 3! \cdot 3! = 72 \end{aligned}$$

weiter ist

$$\gamma\big((0123456)\Gamma_1\big) = \left(\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}; \mathrm{id}\right)$$
$$\Rightarrow \gamma(\Gamma_1) = \left(\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}; (0123456)\right)$$

Da $\widetilde{\Gamma}_1 = \widetilde{\Gamma}_2$ brauchen wir auch hier nur Γ_2 gemäß $\widetilde{\Gamma}_2$ zu sortieren, wir wählen hierzu die Permutation (03)(14)(256):

$$(03)(14)(256)\Gamma_2 = \begin{pmatrix}0 & 0 & 0 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix} \text{ und } \Gamma_2^{min} = \begin{pmatrix}1 & 1 & 1 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 1\end{pmatrix}$$

Es gilt

$$Aut((03)(14)(256)\Gamma_2) = \Big\langle \Big\{ \left(\begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}; (35)(46)\right) \Big\} \cup \{(I_2;\pi_0) \mid \pi_0 \in S_{\{0,1,2\}}\}$$
$$\cup \{(I_2;\pi_1) \mid \pi_1 \in S_{\{3,4\}}\} \cup \{(I_2;\pi_2) \mid \pi_2 \in S_{\{5,6\}}\}\Big\rangle$$
$$\Rightarrow Aut(\Gamma_2) = \big(I_2; (03)(14)(265)\big) \cdot Aut\big((03)(14)(256)\Gamma_1\big) \cdot \big(I_2; (03)(14)(256)\big)$$
$$\Rightarrow |Aut(\Gamma_2)| = 2 \cdot 3! \cdot 2! \cdot 2! = 48$$

weiter ist

$$\gamma\big((03)(14)(256)\Gamma_2\big) = \left(\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}; \mathrm{id}\right)$$
$$\Rightarrow \gamma(\Gamma_2) = \left(\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}; (03)(14)(256)\right)$$

Die Gruppenordnungen $|Aut(\Gamma_1)|$ und $|Aut(\Gamma_2)|$ stimmen mit der Angabe in der Literatur überein.

4.3.2. Die benutzte Homomorphismen auf den weiteren Ebenen

Wir definieren nun auf den weiteren Ebenen $i \in (n+1), i > 0$ die $S_n^{(i)}$–Homomorphismen, welche zum Einsatz kommen. Es sei zu $\Gamma \in GF(q)^{k\times n,k}$ die Matrix $\Gamma^{(i)}$ stets ein i–minimaler Repräsentant der inneren Bahn $\omega(\Gamma)$.

Wir haben bereits in Folgerung 4.1.10 auf Seite 53 den $S_n^{(i)}$–Homomorphismus

$$\Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k} \to GF(q)^{k\times i}$$
$$\omega(\Gamma) \mapsto \Pi_i\big(\Gamma^{(i)}\big)$$

eingeführt.

Wir trennen diesen Test für die einzelnen Spalten $j < i$ auf. Wir können uns im Algorithmus auf die neu festgelegte Spalte $i-1$ beschränken, da die weiteren schon auf den vorhergehenden Ebenen getestet wurden.

4.3.13 Satz (i–Minimalität)**.** *Die Gruppe $S_n^{(i)}$ operiert durch die triviale Zuordnung $\pi \cdot v = v$ auf der revers lexikographisch geordneten Menge $GF(q)^k$. Die Abbildung*

$$f_{i,0} : \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k} \to GF(q)^k$$
$$\omega(\Gamma) \mapsto \Gamma^{(i)}(i-1)$$

ist somit eine $S_n^{(i)}$–Invariante.

Beweis. Die Spalte $i-1$ ist für jeden i–minimalen Repräsentanten einer Bahn identisch, somit ist die Zuordnung unabhängig von Γ und $\Gamma^{(i)}$, die Abbildung wohldefiniert. Für jedes Element $\pi \in S_n^{(i)}$ ist aber $\pi \cdot \Gamma^{(i)}$ auch ein i–minimaler Repräsentant der Bahn $\omega(\pi\Gamma)$. Die Abbildung $f_{i,0}$ ist also ein $S_n^{(i)}$–Homomorphismus. □

4.3.14 Satz (Rank-Refine)**.** *Die Abbildung*

$$f_{i,1} : \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k} \to 2^n$$
$$\omega(\Gamma) \mapsto f_{i,1}(\omega(\Gamma))$$

mit $f_{i,1}(\omega(\Gamma))(j) := \begin{cases} 0, \Gamma^{(i)}(j) \in \left\langle \Gamma^{(i)}(0), \ldots, \Gamma^{(i)}(i-1) \right\rangle \\ 1, \text{sonst} \end{cases}$ *ist wohldefiniert und ein $S_n^{(i)}$–Homomorphismus. Zum Einsatz im Algorithmus wählen wir als Ordnung auf 2^n die lexikographisch aufsteigende Sortierung der Vektoren.*

Beweis. Der i–minimale Repräsentant $\Gamma^{(i)}$ der Bahn $\omega(\Gamma)$ ist unabhängig von Γ. Wir müssen jedoch für zwei verschiedene i–minimale Repräsentanten die Gleichheit der Abbildung zeigen. Es seien also $\Gamma^{(i)}, \widetilde{\Gamma}^{(i)}$ zwei i–minimale Repräsentanten einer Bahn $\omega(\Gamma)$. Da $\Pi_i(\Gamma^{(i)}) = \Pi_i(\widetilde{\Gamma}^{(i)})$ existiert $((A,\varphi);\alpha)) \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(i)})}$ mit

$$\Gamma^{(i)} = ((A,\varphi);\alpha))\widetilde{\Gamma}^{(i)}.$$

Sei nun $j \in n$ mit $\Gamma^{(i)}(j) \in \left\langle \Gamma^{(i)}(0), \ldots, \Gamma^{(i)}(i-1) \right\rangle$ beliebig vorgegeben: Wir haben bereits in Abschnitt 4.1 bewiesen, dass die i–te Projektion eines jeden i–minimalen Repräsentanten $\Gamma^{(i)}$ genau die ersten $\mathrm{Rang}(\Pi_i(\Gamma^{(i)}))$ Einheitsvektoren als Spalten enthält. Für den Beweis bringen wir ein, dass nach Hilfssatz 4.1.15 durch die Matrix A die ersten

$\mathrm{Rang}(\Pi_i(\Gamma^{(i)}))$ Einheitsvektoren auf Vielfache von sich selbst abgebildet werden.

$$\begin{aligned}
&\Gamma^{(i)}(j) \in \left\langle \Gamma^{(i)}(0), \ldots, \Gamma^{(i)}(i-1) \right\rangle \\
\Longleftrightarrow &\forall\ k > l \geq \mathrm{Rang}(\Pi_i(\Gamma^{(i)})) : \Gamma^{(i)}_{l,j} = 0 \\
\Longleftrightarrow &\forall\ k > l \geq \mathrm{Rang}(\Pi_i(\widetilde{\Gamma}^{(i)})) : \widetilde{\Gamma}^{(i)}_{l,j} = 0 \\
\Longleftrightarrow &\widetilde{\Gamma}^{(i)}(j) \in \left\langle \widetilde{\Gamma}^{(i)}(0), \ldots, \widetilde{\Gamma}^{(i)}(i-1) \right\rangle
\end{aligned}$$

Für die Homomorphieeigenschaft der Abbildung nutzen wir wiederum, dass für ein $\pi \in S_n^{(i)}$ auch $\pi \cdot \Gamma^{(i)}$ ein i–minimaler Repräsentant der Bahn $\omega(\pi\Gamma)$ ist. Das Erzeugnis bleibt somit gleich. Es gilt also für ein beliebiges $j \in n$:

$$\begin{aligned}
&f_{i,1}(\pi\omega(\Gamma))(j) = 0 \\
\Longleftrightarrow &(\pi \cdot \Gamma^{(i)})(j) \in \left\langle (\pi \cdot \Gamma^{(i)})(0), \ldots, (\pi \cdot \Gamma^{(i)})(i-1) \right\rangle \\
\Longleftrightarrow &(\Gamma^{(i)})(\pi^{-1}(j)) \in \left\langle \Gamma^{(i)}(0), \ldots, \Gamma^{(i)}(i-1) \right\rangle \\
\Longleftrightarrow &f_{i,1}(\omega(\Gamma))(\pi^{-1}(j)) = 0 \\
\Longleftrightarrow &\big(\pi \cdot f_{i,1}(\omega(\Gamma))\big)(j) = 0
\end{aligned}$$

□

4.3.15 Bemerkung. Die Anwendung des Homomorphismus besagt nichts weiteres, als dass wir diejenigen Spalten innerhalb eines Blocks nach vorne sortieren, welche im Erzeugnis der bisher festgelegten Spalten $\{0, \ldots, i-1\}$ liegen.

4.3.16 Bemerkung. Liegt eine nicht redundante, projektive Generatormatrix vor, so sind je zwei Spalten linear unabhängig. Damit erhalten wir auf Ebene 1 für jede Spalte $j > 0$ den Wert $f_{1,1}(\omega(\Gamma))(j) = 0$.

4.3.17 Folgerung. *Hat sich im Backtrackbaum beim Übergang vom Vater g_i zum Sohn $g_{i+1}, i \in n$ der Rang der entsprechenden Projektionen nicht verändert, dh.*

$$\mathrm{Rang}(\Pi_i(\Gamma^{(g_i,i)})) = \mathrm{Rang}(\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})),$$

so brauchen wir den Test mittels $f_{i+1,1}$ nicht durchzuführen, da $f_{i+1,1} = f_{i,1}$ ist.

4.3.18 Satz (Supp-Refine)**.** *Die Abbildung*

$$\begin{aligned}
f_{i,2} : \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k \times n,k} &\to (2^k \cup \{\infty\})^n \\
\omega(\Gamma) &\mapsto f_{i,2}(\omega(\Gamma))
\end{aligned}$$

mit

$$f_{i,2}(\omega(\Gamma))(j) := \begin{cases} \infty, & f_{i,1}(\omega(\Gamma))(j) = 1 \\ b(j, \Gamma^{(i)}), & \text{sonst} \end{cases}$$

und

$$\forall l \in k : b(j, \Gamma^{(i)})_l := \begin{cases} 0, \Gamma^{(i)}_{j,l} = 0 \\ 1, \text{sonst} \end{cases}$$

ist wohldefiniert und ein $S_n^{(i)}$–Homomorphismus, dabei sei $\infty \notin 2^k$ ein weiteres Element. Um die Menge $2^k \cup \{\infty\}$ anordnen zu können, definieren wir $v < \infty, \forall v \in 2^k$. Die Vektoren $v \in 2^k$ sortieren wir revers lexikographisch. Schließlich seien die Elemente in $(2^k \cup \{\infty\})^n$ ausgehend von dieser Ordnung lexikographisch aufsteigend angeordnet.

Beweis. Wir müssen für die Wohldefiniertheit zeigen, dass für zwei verschieden i–minimale Repräsentanten $\Gamma^{(i)}, \overline{\Gamma}^{(i)}$ einer Bahn $\omega(\Gamma)$, die Funktionswerte an jeder Stelle $j \in n$ gleich sind. Falls der Fall $f_{i,1}(\omega(\Gamma))(j) = 1$ vorliegt, ist die Zuordnung unabhängig von dem Repräsentanten.
Es sei

$$((A, \varphi); \alpha) \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(i)})} : ((A, \varphi); \alpha)\Gamma^{(i)} = \overline{\Gamma}^{(i)}.$$

Da die Einheitsvektoren $\{e_0, \ldots, e_{s-1}\}$ für $s := \text{Rang}(\Pi_i(\Gamma^{(i)}))$ als Spalten in $\Pi_i(\Gamma^{(i)})$ vorkommen, gilt für $l \in s$ und $\widetilde{l} \in i$ sd. $\Gamma^{(i)}(\widetilde{l}) = e_l$:

$$e_l^T = \varphi(\widetilde{l}) A \alpha(e_l^T) = \varphi(\widetilde{l}) A e_l^T.$$

Wir betrachten nun eine Spalte $j \in n$ mit $f_{i,1}(\omega(\Gamma))(j) = 0$:

$$\Rightarrow \forall s \leq l < k : \Gamma^{(i)}_{l,j} = 0 \text{ bzw. } \Gamma^{(i)}(j) = \sum_{l \in s} \mu_l e_l \text{ für geeignete } \mu_l \in GF(q).$$

Weiter gilt

$$\begin{aligned} \overline{\Gamma}^{(i)}(j)^T =& \big(((A, \varphi); \alpha)\Gamma^{(i)}\big)(j)^T = \varphi(j) A \alpha(\Gamma^{(i)}(j)^T) \\ =& \varphi(j) A \alpha\Big(\sum_{l \in s} \mu_l e_l^T\Big) = \varphi(j) \sum_{l \in s} \alpha(\mu_l) A e_l^T \\ =& \sum_{l \in s} \underbrace{\varphi(j)\alpha(\mu_l)\varphi(\widetilde{l})^{-1}}_{\neq 0 \Longleftrightarrow \mu_l \neq 0} e_l^T \end{aligned}$$

und damit

$$b(j, \Gamma^{(i)}) = b(j, \overline{\Gamma}^{(i)}).$$

Für die $S_n^{(i)}$–Homomorphie nutzen wir zunächst, dass $f_{i,1}$ ein $S_n^{(i)}$–Homomorphismus ist. Es gilt also für ein beliebiges $\pi \in S_n^{(i)}$:

$$\begin{aligned} & f_{i,2}(\pi\omega(\Gamma))(j) = \infty \Longleftrightarrow f_{i,1}(\pi\omega(\Gamma))(j) = 1 \Longleftrightarrow f_{i,1}(\omega(\Gamma))(\pi^{-1}(j)) = 1 \\ \Longleftrightarrow & f_{i,2}(\omega(\Gamma))(\pi^{-1}(j)) = \infty \Longleftrightarrow \big(\pi f_{i,2}(\omega(\Gamma))\big)(j) = \infty \end{aligned}$$

Für diejenigen Spalten $j \in n$ mit $f_{i,2}(\pi\omega(\Gamma))(j) \neq \infty$ gilt für ein beliebiges $l \in k$:

$$\begin{aligned}
& f_{i,2}(\pi\omega(\Gamma))(j)_l = b(j, \pi \cdot \Gamma^{(i)})_l = 0 \\
\Longleftrightarrow & (\pi\Gamma^{(i)})_{l,j} = 0 \\
\Longleftrightarrow & (\Gamma^{(i)})_{l,\pi^{-1}(j)} = 0 \\
\Longleftrightarrow & b(\pi^{-1}(j), \Gamma^{(i)})_l = 0 \\
\Longleftrightarrow & \big(\pi \cdot f_{i,2}(\omega(\Gamma))\big)(j)_l = 0
\end{aligned}$$

Zusammenfassend ergibt sich also $f_{i,2}(\pi\omega(\Gamma)) = \pi \cdot f_{i,2}(\omega(\Gamma))$. □

4.3.19 Bemerkung. Durch das zusätzliche Element ∞ erreichen wir, dass nur in den Blöcken gearbeitet wird, die im Erzeugnis der ersten i Spalten liegen. Für die anderen Blöcke können wir dieses Kriterium nicht anwenden, da durch Gaußschritte diese Form noch zerstört werden kann.

4.3.20 Bemerkung. Genauso wie für die Homomorphismen $f_{i,1}$ gilt auch hier für einen Sohn g_{i+1} von g_i:

$$\mathrm{Rang}(\Pi_i(\Gamma^{(g_i,i)})) = \mathrm{Rang}(\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)})) \Rightarrow f_{i,2} = f_{i+1,2}.$$

Wir können also auch hier keine Verfeinerung mittels $f_{i+1,2}$ erreichen.

4.3.21 Hilfssatz. *Es sei $\kappa \in k$ beliebig und $\infty \notin GF(q)^k$ ein weiteres beliebiges Element. Die Abbildungen*

$$\begin{aligned}
g_{i,\kappa} : \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\Big) \backslash\backslash GF(q)^{k\times n,k} &\to \big(GF(q)^k \cup \{\infty\}\big)^n \\
\omega(\Gamma) &\mapsto g_{i,\kappa}\big(\omega(\Gamma)\big)
\end{aligned}$$

mit

$$g_{i,\kappa}\big(\omega(\Gamma)\big)(j) := \begin{cases} \min\big\{\tau^{tx}\big(d(\Gamma^{(i)}, \kappa, j)\big) \mid x \in \{1, \ldots, \frac{r}{t}\}\big\}, & f_{i,1}(\omega(\Gamma))(j) = 0 \wedge \\ & f_{i,2}(\omega(\Gamma))(j)_\kappa = 1 \\ \infty, & \text{sonst} \end{cases}$$

und

$$d(\Gamma^{(i)}, \kappa, j)_l := \begin{cases} \Gamma^{(i)}_{l,j} \cdot {\Gamma^{(i)}_{\kappa,j}}^{-1}, & l \in p_{\widetilde{\kappa}} \\ 0, & \text{sonst} \end{cases}$$

sind wohldefiniert und $S_n^{(i)}$–Homomorphismen, wobei $(s, \mathfrak{p}, t)$ das Stabilisatortripel zu $\Gamma^{(i)}$ sei und $\widetilde{\kappa}$ der eindeutige Index mit $\kappa \in p_{\widetilde{\kappa}}$.

Beweis. Für die Wohldefiniertheit von $g_{i,\kappa}\big(\omega(\Gamma)\big)$ zeigen wir zunächst, dass

$$\min\big\{\tau^{tx}\big(d(\Gamma^{(i)},\kappa,j)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\}$$

unabhängig von dem i–minimalen Repräsentanten $\Gamma^{(i)}$ ist. Sei dazu wie oben

$$((A,\varphi);\alpha) \in \big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q)\big)_{\Pi_i(\Gamma^{(i)})} : ((A,\varphi);\alpha)\Gamma^{(i)} = \overline{\Gamma}^{(i)}.$$

Desweiteren gelte die Voraussetzung für den ersten Fall der Definition von $f_{i,2}$:

$$\begin{aligned}
&f_{i,1}(\omega(\Gamma))(j) = 0 \wedge f_{i,2}(\omega(\Gamma))(j)_\kappa = 1\\
\Rightarrow&\big(\forall s \le l < k : \Gamma^{(i)}_{l,j} = 0\big) \wedge \Gamma^{(i)}_{\kappa,j} \neq 0\\
\Rightarrow&\kappa < s\\
\Rightarrow&\exists\widetilde{\kappa} \text{ sd. } \kappa \in p_{\widetilde{\kappa}}
\end{aligned}$$

Damit ist aber die Inversenbildung ${\Gamma^{(i)}_{\kappa,j}}^{-1}$ und die Definition der Menge $p_{\widetilde{\kappa}}$ zulässig. Im Übrigen sind diese Folgerungen für jeden i–minimalen Repräsentanten möglich und somit insbesondere auch gültig für $\overline{\Gamma}^{(i)}$. Die Mengen $p_{\widetilde{\kappa}}$ sind ebenfalls unabhängig von dem gewählten Repräsentanten.
Analog zu Satz 4.3.18 drücken wir die Spalten mit Hilfe der ersten s Einheitsvektoren aus:

$$\begin{aligned}
&\Gamma^{(i)}(j) = \sum_{\lambda\in s} \mu_\lambda e_\lambda\\
&\overline{\Gamma}^{(i)}(j)^T = \sum_{\lambda\in s} \varphi(j)\alpha(\mu_\lambda)\varphi(\widetilde{\lambda})^{-1} e_\lambda^T\\
\Rightarrow&\Gamma^{(i)}_{l,j} \cdot {\Gamma^{(i)}_{\kappa,j}}^{-1} = \mu_l \cdot \mu_\kappa^{-1} \text{ und}\\
&\overline{\Gamma}^{(i)}_{l,j} \cdot \big(\overline{\Gamma}^{(i)}_{\kappa,j}\big)^{-1} = \big(\varphi(j)\alpha(\mu_l)\varphi(\widetilde{l})^{-1}\big) \cdot \big(\varphi(j)\alpha(\mu_\kappa)\varphi(\widetilde{\kappa})^{-1}\big)^{-1}
\end{aligned}$$

nun ist aber für $l \in p_{\widetilde{\kappa}} : \widetilde{\kappa} = \widetilde{l}$ und somit

$$\overline{\Gamma}^{(i)}_{l,j} \cdot \big(\overline{\Gamma}^{(i)}_{\kappa,j}\big)^{-1} = \alpha(\mu_l \cdot \mu_\kappa^{-1})$$

Da weiter $\alpha = \tau^{ty}$ für ein $y \in \{1,\dots,\frac{r}{t}\}$ und $\alpha(0) = 0$ folgt die Gleichheit der Mengen

$$\big\{\tau^{tx}\big(d(\Gamma^{(i)},\kappa,j)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\} = \big\{\tau^{tx}\big(d(\overline{\Gamma}^{(i)},\kappa,j)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\}.$$

Die Definition von $g_{i,\kappa}\big(\omega(\Gamma)\big)$ ist somit unabhängig von der Wahl des Repräsentanten. Die $S_n^{(i)}$–Homomorphie lässt sich analog zu Satz 4.3.18 durch einfaches Nachrechnen der Definitionen beweisen. □

Wir werden nun als Homomorphismen die $g_{i,\kappa}$ für jedes $\kappa \in k$ rufen.

4.3.22 Satz (Division-Refine)**.** *Die Abbildung*

$$f_{i,3} : \Big((GL_k(q) \times GF(q)^{*n}) \rtimes Gal(q) \Big) \backslash\backslash GF(q)^{k \times n,k} \to \Big(\big(GF(q)^k \cup \{\infty\} \big)^n \Big)^k$$
$$\omega(\Gamma) \mapsto f_{i,3}(\omega(\Gamma))$$

mit

$$f_{i,3}(\omega(\Gamma))(\kappa) := g_{i,\kappa}(\omega(\Gamma)), \kappa \in k$$

ist wohldefiniert und ein $S_n^{(i)}$–Homomorphismus, wenn wir die Operation komponentenweise verstehen.
Als Ordnung in $GF(q)^k \cup \{\infty\}$ setzen wir wiederum $\forall v \in GF(q)^k : v < \infty$ und auf $GF(q)^k$ die revers lexikographische Sortierung. Weiter wählen wir die lexikographische Sortierung für die Vektoren $\big(GF(q)^k \cup \{\infty\}\big)^n$ und schließlich ausgehend von dieser Ordnung die lexikographisch aufsteigende Sortierung für den Bildbereich.

Beweis. Folgt sofort aus der Homomorphieeigenschaft der $g_{i,\kappa}$ und der Eindeutigkeit der ersten i–Spalten eines i–minimalen Repräsentanten. □

4.3.23 Bemerkung. Wir werden auch hier im Algorithmus den Funktionsaufruf komponentenweise durchführen. Für die $\kappa \geq s$ erhalten wir bei dem Aufruf ohnehin stets ∞^n, wir können also keine Spalte von der anderen trennen. Insofern rufen wir nur die Funktionen $g_{i,\kappa}$ für $\kappa < s$.

Bei dem Aufruf der Funktionen $g_{i,\kappa}$ können wir blockweise vorgehen. Liegt ein Block nicht im Erzeugnis ($\iff f_{i,1}(\omega(\Gamma))(j) = 1$), so können wir die Spalten nicht trennen, da für jede Spalte ∞ berechnet wird. Genauso brauchen wir die Berechnung nicht durchzuführen, falls die Einträge in der Zeile κ gleich Null sind. Auch hier erhalten wir für jede Spalte des Blocks den Wert ∞.

4.3.24 Hilfssatz. *Es sei $(s, \mathfrak{p}, t)$ das Stabilisatortripel der ersten i Spalten von $\Gamma^{(i)}$. Wir wählen $\kappa, \lambda < s$ mit $\kappa \neq \lambda$ und $\kappa \in p_{\widetilde{\lambda}}$. Desweiteren liege ein Block $B \subseteq n$ vor, dessen Einträge in den Zeilen κ und λ ungleich Null sind. Für zwei beliebige Spalten $j, l \in B$ gilt*

$$g_{i,\kappa}(\omega(\Gamma))(j) = g_{i,\kappa}(\omega(\Gamma))(l) \iff g_{i,\lambda}(\omega(\Gamma))(j) = g_{i,\lambda}(\omega(\Gamma))(l).$$

Beweis. Für den Fall, dass für eine Spalte $m \in B$ innerhalb des Blocks $f_{i,1}(\omega(\Gamma))(m) = 1$ gelte, nimmt $g_{i,\kappa}(\omega(\Gamma))$ und $g_{i,\lambda}(\omega(\Gamma))$ dort ohnehin nur den Wert ∞ auf allen Spalten an. Dies folgt aus der Tatsache, dass innerhalb eines Blocks, die Bilder der zuvor betrachteten Homomorphismen gerade konstant bleiben.
Es sei also $f_{i,1}(\omega(\Gamma))(m) = 0$ und da die Zeilen als Nichtnullzeilen vorausgesetzt waren

gilt weiter: $f_{i,2}(\omega(\Gamma))(m)_\kappa = 1$ und $f_{i,2}(\omega(\Gamma))(m)_\lambda = 1$ für alle Spalten $m \in B$.
Es bleibt also für den Beweis die folgende Implikation zu zeigen:

$$\begin{aligned}&\big\{\tau^{tx}\big(d(\Gamma^{(i)},\kappa,j)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\} = \big\{\tau^{tx}\big(d(\Gamma^{(i)},\kappa,l)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\}\\ \Rightarrow&\big\{\tau^{tx}\big(d(\Gamma^{(i)},\lambda,j)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\} = \big\{\tau^{tx}\big(d(\Gamma^{(i)},\lambda,l)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\}\end{aligned}$$

Hierzu sei $y \in \{1,\dots,\frac{r}{t}\}$ so dass

$$d(\Gamma^{(i)},\kappa,j) = \tau^{ty}\big(d(\Gamma^{(i)},\kappa,l)\big)$$

gelte und $\mu \in p_{\widetilde{\kappa}}$ eine weitere beliebige Zeile. Wegen der vorausgesetzten Gleichheit für κ gilt weiter:

$$\Gamma^{(i)}_{\mu,j}\cdot {\Gamma^{(i)}_{\kappa,j}}^{-1} = \tau^{ty}\Big(\Gamma^{(i)}_{\mu,l}\cdot {\Gamma^{(i)}_{\kappa,l}}^{-1}\Big) \text{ und } \Gamma^{(i)}_{\lambda,j}\cdot {\Gamma^{(i)}_{\kappa,j}}^{-1} = \tau^{ty}\Big(\Gamma^{(i)}_{\lambda,l}\cdot {\Gamma^{(i)}_{\kappa,l}}^{-1}\Big).$$

Fassen wir beide Gleichungen zusammen so gilt:

$$\begin{aligned}&\Gamma^{(i)}_{\mu,j}\cdot {\Gamma^{(i)}_{\kappa,j}}^{-1}\cdot {\Gamma^{(i)}_{\lambda,j}}^{-1}\cdot \Gamma^{(i)}_{\kappa,j} = \tau^{ty}\Big(\Gamma^{(i)}_{\mu,l}\cdot {\Gamma^{(i)}_{\kappa,l}}^{-1}\Big)\cdot \tau^{ty}\Big(\Gamma^{(i)}_{\lambda,l}\cdot {\Gamma^{(i)}_{\kappa,l}}^{-1}\Big)^{-1}\\ \Rightarrow&\Gamma^{(i)}_{\mu,j}\cdot {\Gamma^{(i)}_{\lambda,j}}^{-1} = \tau^{ty}\Big(\Gamma^{(i)}_{\mu,l}\cdot {\Gamma^{(i)}_{\lambda,l}}^{-1}\Big)\end{aligned}$$

Die Aussage ist wahr für alle $\mu \in p_{\widetilde{\kappa}}$ an den weiteren Stellen sind die Einträge der Vektoren ohnehin gleich 0, es folgt:

$$\begin{aligned}&d(\Gamma^{(i)},\lambda,j) = \tau^{ty}\big(d(\Gamma^{(i)},\lambda,l)\big)\\ \Rightarrow&\big\{\tau^{tx}\big(d(\Gamma^{(i)},\lambda,j)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\} = \big\{\tau^{tx}\big(d(\Gamma^{(i)},\lambda,l)\big) \mid x \in \{1,\dots,\tfrac{r}{t}\}\big\}\end{aligned}$$

□

4.3.25 Folgerung. *Es sei $\kappa < s$ beliebig und $h \in S_n^{(i)}$ so dass die Bilder der vorausgegangenen Homomorphismen minimal sind. Falls für einen Block $B \subseteq n$ der aktuellen kanonischen Younguntergruppe mit $j \in B$ ein Index $\lambda \in \mathrm{supp}((h\cdot\Gamma^{(i)})(j)) \cap p_{\widetilde{\kappa}}$ existiert mit $\lambda < \kappa$, so ist $g_{i,\kappa}(\omega(h\cdot\Gamma^{(i)}))$ auf den Spalten des Blocks konstant. Wir erhalten also keine neue Partitionierung.*

Beweis. Auf den Blöcken der geordneten Mengenpartitionen sind die Bilder der vorausgegangenen Homomorphismen für $h\cdot\Gamma^{(i)}$ konstant. Da bereits der Funktionsaufruf für λ stattgefunden hat, gilt also die Gleichheit der Werte $g_{i,\kappa}(\omega(h\cdot\Gamma^{(i)}))(j), j \in B$. □

4.3.26 Bemerkung. Aufgrund der Folgerung brauchen wir also für jeden Block B die Partitionierung mittels $g_{i,\kappa}$ nur dann vorzunehmen, falls für eine beliebige Spalte $j \in B$ gilt: $\kappa = \min\{\lambda \in p_{\widetilde{\kappa}} \mid (h\cdot\Gamma^{(i)})_{\lambda,j} \neq 0\}$.

4.3.27 Bemerkung. Wir wollen wieder ein Kriterium festlegen, bei denen die Funktionen $g_{i+1,\kappa}(\omega(g_{i+1}\Gamma))$ im Vergleich zum Vaterknoten g_i keine neuen Erkenntnisse liefern. Es gilt:

$$t^{(i+1)} = t^{(i)} \wedge p_{\widetilde{\kappa}}^{(i+1)} \in \mathfrak{p}^{(i)} \Rightarrow g_{i,\kappa}(\omega(g_i\Gamma)) = g_{i+1,\kappa}(\omega(g_i\Gamma))$$

Da g_{i+1} aus g_i ohnehin nur durch Vertauschen von Spalten hervorgeht, welche den selben Wert für $g_{i,\kappa}(\omega(g_i\Gamma))$ annehmen, gilt also auch:

$$t^{(i+1)} = t^{(i)} \wedge p_{\widetilde{\kappa}}^{(i+1)} \in \mathfrak{p}^{(i)} \Rightarrow g_{i,\kappa}(\omega(g_i\Gamma)) = g_{i+1,\kappa}(\omega(g_{i+1}\Gamma))$$

Außerdem ist der Aufruf von $g_{i,\kappa}$ für $|p_{\widetilde{\kappa}}| = 1$ nicht zum Partitionieren geeignet, da in diesem Fall als Bilder der Spalten entweder der Einheitsvektor e_κ oder ∞ berechnet wird. Diese Unterscheidung ist jedoch nur von den Werten von $f_{i,1}$ und $f_{i,2}$ abhängig und somit bereits konstant auf den Blöcken.

Zusammenfassend gilt hier im Gegensatz zu den Homomorphismen $f_{i,1}$ und $f_{i,2}$ für einen Sohn g_{i+1} von g_i:
Ist $\mathrm{Rang}(\Pi_i(\Gamma^{(g_i,i)})) \neq \mathrm{Rang}(\Pi_{i+1}(\Gamma^{(g_{i+1},i+1)}))$ so können wir keine Verfeinerung mittels $f_{i+1,3}$ erreichen.

4.3.3. Weitere Homomorphismen

In der Implementierung des Softwarepakets konnten nur einige Homomorphismen eingebracht werden. Diese wurden in den vorausgegangen Abschnitten ausführlich erklärt. An dieser Stelle möchten wir noch kurz anführen, wie wir aus einer Invarianten für Codes geeignete Homomorphismen für unseren Algorithmus erzeugen. Wir übertragen hierzu die Formulierungen aus [Sen00], die aber dort auf den Spezialfall der Permutationsisometrien eingeschränkt wurden.

4.3.28 Definition. Es sei $\mathcal{L}$ die Menge der linearen Codes. Eine Abbildung

$$\mathcal{V} : \mathcal{L} \to F$$

in eine Menge F heißt Codeinvariante über F, falls für alle semilinear isometrischen Codes C, C' die Funktionswerte $\mathcal{V}(C), \mathcal{V}(C')$ gleich sind.

4.3.29 Beispiel. Beispiele für Codeinvarianten sind die Zuweisung des Codes auf seine Minimaldistanz oder Gewichtsverteilung.

4.3.30 Hilfssatz. *Es sei $i \in (n+1)$ beliebig und $\Gamma^{(i)}$ ein beliebiger i-minimaler Repräsentant eines Knotens auf Ebene i. Weiter sei $(s, \mathfrak{p}, t)$ das Stabilisatortripel zu den ersten i Spalten von $\Gamma^{(i)}$. Dann ist $\Gamma^{(i)}$ von folgender Form*

$$\Gamma^{(i)} = \begin{pmatrix} \Gamma_0 & \Gamma_1 \\ 0 & \Gamma_2 \end{pmatrix}$$

mit $\Gamma_0 \in GF(q)^{s\times i}, \Gamma_1 \in GF(q)^{s\times(n-i)}$ *und* $\Gamma_2 \in GF(q)^{(k-s)\times(n-i)}$*. Es gilt für die Punktierung bzw. Verkürzung an den ersten* i *Koordinatenpositionen:*

$$\left(C(\Gamma^{(i)})\right)_{\{0,\dots,i-1\}} = C\left(\begin{pmatrix}\Gamma_1\\ \Gamma_2\end{pmatrix}\right) \text{ und } \left(C(\Gamma^{(i)})\right)_{\setminus\{0,\dots,i-1\}} = C(\Gamma_2)$$

Beweis. Die Gestalt der Matrix folgt aus der i-Minimalität. Die Aussage für die Punktierung ist klar. Um den verkürzten Code zu erhalten, ist es notwendig alle Codevektoren auszuwählen, die an den ersten i Spalten nur aus Nulleinträgen bestehen. Diese werden aber gerade von der Matrix $\begin{pmatrix}0 & \Gamma_2\end{pmatrix}$ erzeugt. Anschließend werden die ersten i Spalten gestrichen, es bleibt Γ_2 als Generatormatrix des verkürzten Codes. □

4.3.31 Bemerkung. An dieser Stelle erklärt sich auch die leicht abweichende Definition der Verkürzung.

4.3.32 Hilfssatz. *Es sei* $\mathcal{V}$ *eine Codeinvariante über einer Menge* F*. Wir gewinnen aus* $\mathcal{V}$ *den* $S_n^{(i)}$*-Homomorphismus*

$$\begin{aligned} f :\Big((GL_k(q)\times GF(q)^{*n})\rtimes Gal(q)\Big)\backslash\backslash GF(q)^{k\times n,k} &\to (F\cup\{\infty\})^n\\ \omega(\Gamma) &\mapsto f(\omega(\Gamma)) \end{aligned}$$

mit $\infty \notin F$ *einem beliebigem weiteren Symbol und*

$$f(\omega(\Gamma))_j := \begin{cases}\infty, & j<i\\ \mathcal{V}(\widetilde{C}_{j-i}), & i\le j<n\end{cases}$$

wobei wir für $\widetilde{C}$ *entweder* $\left(C(\Gamma^{(i)})\right)_{\{0,\dots,i-1\}}$ *oder* $\left(C(\Gamma^{(i)})\right)_{\setminus\{0,\dots,i-1\}}$ *setzen können.*

Beweis. Es sei $i\le j<n$ eine beliebige Spalte. Wir beweisen die Aussage nur für den Fall $\left(C(\Gamma^{(i)})\right)_{\setminus\{0,\dots,i-1\}}$. Die Abbildung ist wohldefiniert, da für einen weiteren i-minimalen Repräsentanten $\widetilde{\Gamma}^{(i)}\in\omega(\Gamma^{(i)})$ gilt:

$$\exists B\in GL_{k-s}(q), \psi\in GF(q)^{*n-i}, \alpha\in Gal(q) \text{ sd. } B\cdot\alpha(\widetilde{\Gamma}_2^{(i)})\cdot M^T_{(\psi;\mathrm{id}_{S_{n-i}})} = \Gamma_2^{(i)}$$

Streichen wir aus ψ den Eintrag $(j-i)$, so gilt:

$$\begin{aligned} &\exists\overline{\psi}\in GF(q)^{*n-i-1}, \alpha\in Gal(q) \text{ sd.}\\ &(\overline{\psi};(\alpha,\mathrm{id}_{S_{n-i-1}}))\Big(\left(C(\widetilde{\Gamma}^{(i)})\right)_{\setminus\{0,\dots,i-1\}}\Big)_{j-i} = \Big(\left(C(\Gamma^{(i)})\right)_{\setminus\{0,\dots,i-1\}}\Big)_{j-i}\\ \Rightarrow&\mathcal{V}\left(\Big(\left(C(\widetilde{\Gamma}^{(i)})\right)_{\setminus\{0,\dots,i-1\}}\Big)_{j-i}\right) = \mathcal{V}\left(\Big(\left(C(\Gamma^{(i)})\right)_{\setminus\{0,\dots,i-1\}}\Big)_{j-i}\right) \end{aligned}$$

Damit ist die Zuordnung unabhängig von dem gewählten i-minimalen Repräsentanten. Eine Permutationen $\pi\in S_n^{(i)}$ induziert eine Permutation $\overline{\pi}\in S_{n-i}$ der Spalten von $\Gamma_2^{(i)}$.

Streichen wir nach dem Permutieren die Spalte $j-i$ so erhalten wir eine Code der permutationsisometrisch ist zu dem Code, der entsteht, wenn wir die Spalte $\overline{\pi}^{-1}(j-i)$ aus $\Gamma_2^{(i)}$ streichen. Da $\mathcal{V}$ eine Codeinvariante ist, muss aber $f(\omega(\pi\Gamma))_j = f(\omega(\Gamma))_{\pi^{-1}(j)}$ gelten. □

4.3.33 Bemerkung. Wir können als Indexmenge J zum Punktieren bzw. Verkürzen nicht nur $\{0,\ldots,i-1\}$ wählen. Vielmehr ist an dieser Stelle eine beliebige Vereinigung der Blöcke der kanonischen Younguntergruppen geeignet, eine Verfeinerung zur Verfügung zu stellen. Somit können wir diesen Typ von $S_n^{(i)}$-Homomorphismus iterativ anwenden, bis wir alle Möglichkeiten ausgeschöpft haben. Dieses Verfahren nennt man auch das Prinzip der Iterierten Verfeinerung, vgl. [Gug05].

4.4. Kanonisierung des dualen Codes

Zu einem linearen (n,k) – Code sind die Parameter des dualen Codes $(n,n-k)$. Da die Laufzeit des Algorithmus vor allem von der Anzahl der Zeilen der Matrix abhängig ist, siehe Anhang A.1, ist es sinnvoll bei linearen Codes mit $k > \frac{n}{2}$ die Berechnung für die Kontrollmatrix durchzuführen. Eine kanonische Generatormatrix erhält man dann durch eine fest gewählte Zuordnung

$$dual : GF(q)^{n-k\times n,n-k} \to GF(q)^{k\times n,k}, \Delta \mapsto \Gamma$$

4.4.1 Hilfssatz. *Für die monomiale Matrix $M_{(\varphi;\pi)}$ eines $(\varphi;\pi) \in GF(q)^* \wr_n S_n$ gilt:*

$$M_{(\varphi;\pi)}^{-1} = M_{(\varphi_{\pi^{-1}}^{-1};\pi^{-1})} \tag{4.1}$$

$$M_{(\varphi;\pi)}^{T} = M_{(\varphi_{\pi^{-1}};\pi^{-1})} \tag{4.2}$$

$$(M_{(\varphi;\pi)}^{T})^{-1} = (M_{(\varphi;\pi)}^{-1})^{T} = M_{(\varphi^{-1};\pi)} \tag{4.3}$$

Beweis. (4.1) folgt aus der Inversenbildung in $GF(q)^* \wr_n S_n$ und dem Isomorphismus zwischen $GF(q)^* \wr_n S_n$ und $M_n(q)$.
Die Gleichung (4.2) folgt aus:

$$M_{(\varphi;\pi)}^{T} = \left(\begin{array}{c} \\ j=\pi(i) \\ \\ \\ \\ \\ \\ \\ \end{array} \left(\begin{array}{ccccccc} & & & i & & & \\ & & & 0 & & & \\ & & & \vdots & & & \\ & & & 0 & & & \\ 0 & \ldots & 0 & \varphi(j) & 0 & \ldots & 0 \\ & & & 0 & & & \\ & & & \vdots & & & \\ & & & 0 & & & \end{array} \right) \right)^{T} =$$

$$
\begin{array}{c} \\ i \end{array}
\begin{array}{c}
\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}
\begin{pmatrix} & & & & & & \\ & & & & & & \\ & & & & & & \\ 0 & \dots & 0 & \varphi(j) & 0 & \dots & 0 \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{pmatrix}
\end{array}
=
$$

$$
\begin{array}{c} \\ i = \pi^{-1}(j) \end{array}
\begin{array}{c}
\begin{array}{c} j \\ 0 \\ \vdots \\ 0 \\ \\ \\ \end{array}
\begin{pmatrix} & & & & & & \\ & & & & & & \\ & & & & & & \\ 0 & \dots & 0 & \varphi(\pi(i)) & 0 & \dots & 0 \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{pmatrix}
\end{array}
= M_{(\varphi_{\pi^{-1}};\pi^{-1})}
$$

Wir wenden nun (4.1) und (4.2) an um (4.3) zu beweisen:

$$(M^T_{(\varphi;\pi)})^{-1} = M_{(\varphi_{\pi^{-1}};\pi^{-1})}{}^{-1} = M_{\left(\left(\varphi_{\pi^{-1}}\right)_\pi^{-1};(\pi^{-1})^{-1}\right)} = M_{(\varphi^{-1};\pi)}$$

und

$$(M^{-1}_{(\varphi;\pi)})^T = M_{(\varphi^{-1}_{\pi^{-1}};\pi^{-1})}{}^T = M_{\left(\left(\varphi^{-1}_{\pi^{-1}}\right)_\pi;(\pi^{-1})^{-1}\right)} = M_{(\varphi^{-1};\pi)}$$

□

4.4.2 Satz. *Sei Δ eine Kontrollmatrix des (n,k) – Codes C und Γ eine Generatormatrix. Weiter sei $\big((A,\varphi);(\alpha,\pi)\big) \in (GL_{n-k}(q) \times GF(q)^{*n-k}) \rtimes (Gal(q) \times S_n)$ beliebig. Dann ist:*

$$\widetilde{\Gamma} := \big((I_k,\varphi^{-1});(\alpha,\pi)\big)\Gamma$$

eine Generatormatrix des durch die Kontrollmatrix

$$\widetilde{\Delta} := \big((A,\varphi);(\alpha,\pi)\big)\Delta$$

eindeutig bestimmten Codes $\widetilde{C}$.

Beweis.

$$\begin{aligned}\widetilde{\Gamma}\cdot\widetilde{\Delta}^T &= I_k\cdot\alpha(\Gamma)\cdot {M_{(\varphi^{-1};\pi)}}^T\cdot\big(A\cdot\alpha(\Delta)\cdot {M_{(\varphi;\pi)}}^T\big)^T\\ &= \alpha(\Gamma)\cdot {M_{(\varphi^{-1};\pi)}}^T\cdot M_{(\varphi;\pi)}\cdot\alpha(\Delta)^T\cdot A^T\\ &= \alpha(\Gamma)\cdot {M_{(\varphi;\pi)}}^{-1}\cdot M_{(\varphi;\pi)}\cdot\alpha(\Delta^T)\cdot A^T\\ &= \alpha(\Gamma\cdot\Delta^T)\cdot A^T\\ &= \alpha(0_{k\times(n-k)})\cdot A^T = 0_{k\times(n-k)}\end{aligned}$$

$\Rightarrow \widetilde{\Gamma}$ ist eine Generatormatrix von $\widetilde{C}$. □

4.4.3 Folgerung. *Ist $\overline{\Gamma}$ eine weitere beliebige Generatormatrix von $\widetilde{C}$, so gibt es ein eindeutig bestimmtes $B\in GL_k(q)$ mit $\big((B,\varphi^{-1});(\alpha,\pi)\big)\Gamma=\overline{\Gamma}$*

Beweis. Wenden wir obigen Satz an, so ist $\widetilde{\Gamma}$ eine Generatormatrix von $\widetilde{C}$. Also unterscheidet sich $\widetilde{\Gamma}$ und $\overline{\Gamma}$ nur durch Linksmultiplikation einer invertierbaren $k\times k$–Matrix B. Die Matrix ist weiterhin eindeutig bestimmt, da $\widetilde{\Gamma}$ vollen Zeilenrang k hat.

$$\overline{\Gamma}=B\cdot\widetilde{\Gamma}=B\cdot\Big(\big((I_k,\varphi^{-1});(\alpha,\pi)\big)\Gamma\Big)=\big((B,\varphi^{-1});(\alpha,\pi)\big)\Gamma.$$

□

4.4.4 Folgerung. *Ist $\big((A,\varphi);(\alpha,\pi)\big)$ ein Automorphismus von Δ, so existiert ein eindeutig bestimmtes $B\in GL_k(q)$ so dass $\big((B,\varphi^{-1});(\alpha,\pi)\big)$ ein Automorphismus von Γ ist.*

Beweis. Es ist $\widetilde{C}=C$, setze nun $\overline{\Gamma}=\Gamma$. □

4.4.5 Folgerung. *Es sei $Aut(\Delta)$ die Automorphismengruppe zu Δ. Dann ist*

$$\begin{aligned}Aut(\Gamma)=\big\{\big((B,\varphi^{-1});(\alpha,\pi)\big)\mid{}&\big((A,\varphi);(\alpha,\pi)\big)\in Aut(\Delta)\wedge\\ &B\in GL_k(q)\ sd.\ \big((B,\varphi^{-1});(\alpha,\pi)\big)\Gamma=\Gamma\big\}.\end{aligned}$$

Insbesondere gilt damit $|Aut(\Gamma)|=|Aut(\Delta)|$.

4.4.6 Hilfssatz. *Es sei $k>\frac{n}{2}$ und T eine Transversale von*

$$\Big((GL_{n-k}(q)\times GF(q)^{*n-k})\rtimes(Gal(q)\times S_n)\Big)\backslash\backslash GF(q)^{(n-k)\times n,n-k}.$$

Dann ist $dual(T)$ eine Transversale von

$$\Big((GL_k(q)\times GF(q)^{*k})\rtimes(Gal(q)\times S_n)\Big)\backslash\backslash GF(q)^{k\times n,k}.$$

Beweis. Aus dem Satz 4.4.2 folgt sofort, dass für zwei Generatormatrizen semilinear isometrischer Codes auch die Kontrollmatrizen semilinear isometrische Codes erzeugen. Aus der Dualität folgt die Umkehrung der Aussage. Damit ist aber $dual(T)$ ein vollständiges und zum anderen minimales Repräsentantensystem der Bahnen, also eine Transversale der Operation. □

Für die Generatormatrizen $\Gamma \in GF(q)^{k\times n,k}$ mit $k > \frac{n}{2}$ rufen wir also nicht direkt den Kanonisierungsalgorithmus. Vielmehr berechnen wir eine Kontrollmatrix Δ und rufen für diese den Algorithmus. Es sei Δ^{opt} die kanonische Geneneratormatrix aller zu $C^{\perp}$ semilinear isometrischen Codes und $\big((A,\varphi);(\alpha,\pi)\big)\Delta = \Delta^{opt}$. Wir setzen $\Gamma^{opt} = dual(\Delta^{opt})$ und berechnen das eindeutig bestimmte $B \in GL_k(q)$, so dass $\big((B,\varphi^{-1});(\alpha,\pi)\big)\Gamma = \Gamma^{opt}$. Die Bestimmung der Automorphismengruppe $Aut(\Gamma)$ ist dann über Folgerung 4.4.5 leicht möglich.

4.5. Der Algorithmus zur Kanonisierung von Generatormatrizen

Die Verwaltung der Stabilisatoren, allesamt kanonischen Younguntergruppen, erfolge durch die entsprechenden Zahlpartitionen von n. Aus diesem Grunde können wir die Transversale aus Abschnitt 4.2 verwenden. Für $l < i-1$ gilt $\alpha_l^{(i-1)} = 1$ und somit ist $\alpha_{i-1}^{(i-1)}$ die Ordnung des Blocks, welcher die Spalte $(i-1)$ enthält. Die Söhne der Permutation g_{i-1} erreichen wir, indem wir jede Spalten des Blocks jeweils an Position $(i-1)$ bringen.

Die Funktion *minimize* minimiere gemäß Abschnitt 4.1 die Spalte $i-1$ der vorliegenden Matrix unter dem Stabilisator der vorausgegangenen Spalten. Dabei werde auch das Stabilisatortripel $(s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)})$ der ersten i Spalten zu dem i–minimalen Repräsentanten $\Gamma^{(i)}$ berechnet.

Da der Aufruf des $S_n^{(i)}$–Homomorphismus $f_{i,0}$ keine verfeinerte Partitionierung der Spalten ergibt, wurde dieser Funktionsaufruf in Algorithmus 4.2 gesondert mittels $\Gamma^{(i)}(i-1) < \Gamma^{opt}(i-1)$ ausgeführt. Im Funktionsaufruf $refine$ seien die weiteren, in Abschnitt 4.3.2 aufgeführten Homomorphismen zusammengefasst.

4.5.1 Bemerkung. Um eine leichte Verständlichkeit des Algorithmus 4.2 zu erzielen, wurden in den Zeilen 15 und 20 Tests zusammengefasst, welche schon parallel zu den Berechnungen durchgeführt werden sollten.

4.5.2 Beispiel. Wir wollen nun anhand eines Beispiels über $GF(16)$ die Arbeitsweise des Algorithmus nochmals aufzeigen. Es sei $\langle\xi\rangle = GF(16)^*$ ein beliebiges primitives Element und $0 < 1 < \xi < \xi^2 < \ldots < \xi^{14}$ und $\xi^4 = \xi^3 + 1$. Wir starten auf Ebene 3 mit

Algorithmus 4.2 Kanonisierer für nicht redundante Generatormatrizen gemäß Abschnitt 4.3.1

Input: $\Gamma^{(0)} := \Gamma \in GF(q)^{k \times n,k}$
Input: $\alpha^{(0)} := \alpha = [\alpha_0^{(0)}, \ldots, \alpha_{m-1}^{(0)}]$ Zahlpartition von n (Backtracking zu $S_{[\alpha_0^{(0)},\ldots,\alpha_{m-1}^{(0)}]}$)
(z.B. erreicht durch die Reduzierung der Spalten gemäß Homomorphismen $f_{0,1}$ und $f_{0,2}$)
Input: S Gruppe von Automorphismen als vollständiges Labelled Branching
Output: $g \in S_{[\alpha_0^{(0)},\ldots,\alpha_{m-1}^{(0)}]}$ Permutation, sd. $\omega(g\Gamma)$ minimal
Output: $\langle S \rangle = Aut(\omega(\Gamma))$

1: $g_{opt} \leftarrow$ id;
2: $(s^{(0)}, \mathfrak{p}^{(0)}, t^{(0)}) \leftarrow (0, \{\emptyset\}, 1)$;
3: **while** Durchlaufe Backtrackbaum ohne Wurzel **do**
4: $\quad i \leftarrow$ Ebene des aktuellen Knotens;
5: $\quad j \leftarrow$ Index der eingehenden Kante; // $j < \alpha_{i-1}^{(i-1)}$
6: $\quad \Gamma^{(i)} \leftarrow (i-1, i-1+j)\Gamma^{(i-1)}$;
7: $\quad g_i \leftarrow (i-1, i-1+j)g_{i-1}$;
8: $\quad l \leftarrow g_i^{-1}(i-1)$;
9: $\quad$ **if** $S.father_l \geq 0 \wedge g_i(S.father_l) \geq i-1$ **then**
10: $\quad\quad$ **continue on level** i;
11: $\quad$ **end if**
12: $\quad (s^{(i)}, \mathfrak{p}^{(i)}, t^{(i)}, \Gamma^{(i)}) \leftarrow minimize(i, s^{(i-1)}, \mathfrak{p}^{(i-1)}, t^{(i-1)}, \Gamma^{(i)})$;
13: $\quad \alpha^{(i)} \leftarrow \alpha^{(i-1)} \cap S_n^{(i)}$;
14: $\quad \left(flag, h, \alpha^{(i)}\right) \leftarrow refine(i, \Gamma^{(i)}, \alpha^{(i)}, \Gamma^{opt})$;
15: $\quad$ **if** $\Gamma^{(i)}(i-1) > \Gamma^{opt}(i-1) \vee (\Gamma^{(i)}(i-1) = \Gamma^{opt}(i-1) \wedge flag = bigger)$ **then**
16: $\quad\quad$ **continue on level** i; // Überspringe darunterliegenden Teilbaum
17: $\quad$ **else**
18: $\quad\quad g_i \leftarrow hg_i$;
19: $\quad\quad \Gamma^{(i)} \leftarrow h\Gamma^{(i)}$; // $h \in S_n^{(i)}$ zur Minimierung der Bilder der Homomorphismen
20: $\quad\quad$ **if** $\Gamma^{(i)}(i-1) < \Gamma^{opt}(i-1) \vee (\Gamma^{(i)}(i-1) = \Gamma^{opt}(i-1) \wedge flag = smaller)$ **then**
21: $\quad\quad\quad g_{opt} \leftarrow g_i$; // Ein neuer Kandidat für das kanonisierende Element
22: $\quad\quad\quad \Gamma^{opt} \leftarrow \Gamma^{(i)}$;
23: $\quad\quad$ **end if**
24: $\quad\quad$ **if** $i = n$ **then**
25: $\quad\quad\quad S.sift(g_{opt}^{-1}g_i)$; // Ein neuer Automorphismus, möglicherweise auch id
26: $\quad\quad\quad S.complete()$;
27: $\quad\quad\quad$ **continue on level** i;
28: $\quad\quad$ **else**
29: $\quad\quad\quad$ **continue on level** $i+1$;
30: $\quad\quad$ **end if**
31: $\quad$ **end if**
32: **end while**
33: **return** (g_{opt}, S);

Algorithmus 4.3 $refine(i, \Gamma, \alpha, \Gamma^{opt})$ – Verfeinerer gemäß Homomorphieprinzip

Input: $i \in \{1, \ldots, n\}$, weiter sei j_i die Anzahl der Homomorphismen auf Ebene i bekannt.
Input: Γ i–minimal
Input: $\alpha = [\alpha_0, \ldots, \alpha_{m-1}]$ Zahlpartition von n zur Beschreibung des Stabilisators $S_{[\alpha_0,\ldots,\alpha_{m-1}]}$
Input: Γ^{opt} weitere Matrix zum Vergleichen der Bilder
Output: $h \in S_{[\alpha_0,\ldots,\alpha_{m-1}]}$ Permutation, sd. $\overline{f}_i(\omega(\Gamma))$ minimal in der Bahn unter $S_{[\alpha_0,\ldots,\alpha_{m-1}]}$
Output: β Verfeinerung von α zur Beschreibung des Stabilisators von $\overline{f}_i(\omega(\Gamma))$ in $S_{[\alpha_0,\ldots,\alpha_{m-1}]}$

1: $\beta \leftarrow \alpha$;
2: $h \leftarrow \mathrm{id}$;
3: $flag \leftarrow equal$;
4: **for** $j \leftarrow 1$ **to** $j_i - 1$ **do**
5: $\quad (g, \beta) \leftarrow can_{Y_{i,j}}(\beta, f_{i,j}(\omega(\Gamma)))$;
6: $\quad h \leftarrow gh$;
7: $\quad \Gamma \leftarrow g\Gamma$;
8: $\quad$ **if** $flag \neq smaller \wedge f_{i,j}(\omega(\Gamma)) > f_{i,j}(\omega(\Gamma^{opt}))$ **then**
9: $\quad\quad$ **return** $(bigger, \mathrm{id}, \alpha)$; // nur die Rückgabe *bigger* ist von Bedeutung
10: $\quad$ **else**
11: $\quad\quad$ **if** $flag = equal \wedge f_{i,j}(\omega(\Gamma)) < f_{i,j}(\omega(\Gamma^{opt}))$ **then**
12: $\quad\quad\quad flag \leftarrow smaller$;
13: $\quad\quad$ **end if**
14: $\quad$ **end if**
15: **end for**
16: **return** $(flag, h, \beta)$;

dem 3–minimalen Repräsentanten

$$\Gamma^{(3)} := \left(\begin{array}{c|c|c|cc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \xi^{14} & \xi^{5} & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

und Stabilisatortripel $(2, \{0, 1\}, 1)$, da erst hier die Homomorphismen aus Abschnitt 4.3.2 greifen. Weiter sei noch $\Gamma^{opt} = \Gamma^{(3)}$ und $g_3 = g_{opt} = \mathrm{id}$.

Da der Rang der festgelegten Spalten gleich geblieben ist, können wir den Aufruf der Funktionen $f_{3,1}$ und $f_{3,2}$ überspringen. Nun rufen wir die Funktion $g_{3,0}(\omega(\Gamma^{(3)}))$ zunächst nur für den Block $\{3, 4\}$ auf.

$$g_{3,0}(\omega(\Gamma^{(3)}))(3) = \begin{pmatrix} 1 \\ \xi^7 \\ 0 \end{pmatrix} \text{ und } g_{3,0}(\omega(\Gamma^{(3)}))(4) = \begin{pmatrix} 1 \\ \xi^5 \\ 0 \end{pmatrix}.$$

Die beiden Spalten werden also durch den Homomorphismus getrennt. Außerdem müssen wir die Spalten vertauschen, um im Bildbereich den minimalen Repräsentanten der Bahn zu erhalten. Da bereits

$$g_{3,0}(\omega(\Gamma^{(3)}))(5) = \infty$$

gilt, können wir die weiteren Berechnungen für den Block $\{5, 6, 7\}$ bereits an dieser Stelle abbrechen. Die Homomorphismen $g_{3,1}$ und $g_{3,2}$ führen zu keiner weiteren Veränderung, somit erhalten wir als Rückgabe des Aufrufs von $refine$:

$$(smaller, (34), [1, 1, 1, 1, 1, 3]) \leftarrow refine(3, \Gamma^{(3)}, [1, 1, 1, 2, 3], \Gamma^{opt})$$

bzw. in Matrixschreibweise

$$\Gamma^{(3)} \leftarrow (34) \cdot \Gamma^{(3)} = \left(\begin{array}{c|c|c|c|c|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \xi^5 & \xi^{14} & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right).$$

Wir ersetzen die optimalen Kandidaten und gehen im Backtrackbaum eine Ebene tiefer. Auf Ebene 4 ist nur ein Sohn zu betrachten, da der Block, welcher die Koordinate $3 = 4 - 1$ enthält, einelementig ist. Eine Minimierung der Spalte gemäß Algorithmus 4.1 führt zu

$$\Gamma^{(4)} \leftarrow \left(\begin{array}{c|c|c|c|c|ccc} 1 & 0 & 1 & \xi^5 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & \xi^{13} & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

und

$$(s^{(4)}, \mathfrak{p}^{(4)}, t^{(4)}) \leftarrow (2, \{0, 1\}, 2).$$

Der Aufruf von $refine$ ergibt keine Veränderung, jedoch müssen wir Γ^{opt} ersetzen.

Auch auf Ebene 5 ist nur ein Sohn zu betrachten. Es gilt nach der Minimierung:

$$\Gamma^{opt} \leftarrow \Gamma^{(5)} \leftarrow \left(\begin{array}{c|c|c|c|c|ccc} 1 & 0 & 1 & \xi^5 & \xi^2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

und

$$(s^{(5)}, \mathfrak{p}^{(5)}, t^{(5)}) \leftarrow (2, \{0, 1\}, 4).$$

Auf Ebene 6 erhalten wir drei Söhne. Für den Index $j = 0$ ist dies $g_6 = \mathrm{id} g_5 = (34)$. Durch den Aufruf $\alpha^{(i)} \leftarrow \alpha^{(i-1)} \cap S_n^{(i)}$ wird diese Koordinate von den beiden weiteren

getrennt. Der Aufruf der Funktion *minimize* bewirkt hier einen Gaußschritt, d.h. wir bringen die Spalte auf den Einheitsvektor e_2^T:

$$\Gamma^{(6)} \leftarrow \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

und

$$(s^{(6)}, \mathfrak{p}^{(6)}, t^{(6)}) \leftarrow (3, \{\{0,1\},\{2\}\}, 4).$$

Der Aufruf von $refine$ führt bereits im ersten Schleifendurchlauf mittels der Funktion $f_{6,1}$ zu einer Trennung der beiden verbliebenen Spalten, sowie der zur Minimierung des Bildes notwendigen Transposition (67). Wir ersetzen

$$\Gamma^{(6)} \leftarrow (67) \cdot \Gamma^{(6)} = \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right).$$

Aufgrund des kleineren Bildes ersetzen wir $\Gamma^{opt} \leftarrow \Gamma^{(6)}$ und $g_{opt} \leftarrow (67)(34)$. Die Ebenen 7 und 8 führen zu keiner Veränderung der Matrix $\Gamma^{(6)}$. Wir springen im Backtrackverfahren zurück zum nächsten, unbetrachteten Knoten. Dieser liegt auf Ebene 6 und hat als Index der eingehenden Kante $j = 1$. Wir haben also die Spalten 5 und 6 der Matrix $\Gamma^{(5)}$ zu vertauschen und die Spalte 5 abzutrennen.

$$\Gamma^{(6)} \leftarrow \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

der Gausschritt führt zu

$$\Gamma^{(6)} \leftarrow \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right).$$

Der Aufruf von $refine$ ergibt im ersten Durchlauf eine notwendige Vertauschung der beiden letzten Spalten. Da aber das Bild unter dem Homomorphismus $f_{6,2}$ der nun vorletzten Spalte im Vergleich zu Γ^{opt} größer ist, erhalten wir als Rückgabe von $refine$ $flag = bigger$ und können deshalb die Betrachtung an dieser Stelle abbrechen.
Wir gehen zum nächsten Sohn auf Ebene 6 mit Eingang $j = 2$:

$$\Gamma^{(6)} \leftarrow (57) \cdot \Gamma^{(5)} = \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

der Gausschritt führt zu dem 6–minimalen Repräsentanten

$$\Gamma^{(6)} \leftarrow \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right).$$

Auch hier gibt der Aufruf von $refine$ wieder an, die letzten beiden Spalten zu vertauschen. Jedoch ergibt sich anschließend, dass das Bild unter Homomorphismus $f_{6,2}$ dann günstiger ist, und wir erhalten zusätzlich die Rückgabe $smaller$. Somit müssen wir den optimalen Kandidaten ersetzen:

$$\Gamma^{opt} \leftarrow \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & 0 & 1 & \xi^5 & \xi^2 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right) \text{ und } g_{opt} \leftarrow (67)(57)(34) = (34)(567).$$

Die Minimierung auf den weiteren Ebenen 7 und 8, besteht nur darin, die letzten beiden Spalten zu normieren. Dies ist aber bereits der Fall, somit bleibt der Repräsentant und damit auch Γ^{opt} gleich.

Der Teilbaum unterhalb von $\Gamma^{(3)}$ ist somit vollständig abgearbeitet, wir würden mit dem nächsten Bruder von $\Gamma^{(3)}$ fortfahren.

5. Das Sotwarepaket „CodeCan“

Zur Implementierung des Kanonisierers wurde die Programmiersprache C++ gewählt. Dies erfolgt vor allem aus der Tatsache heraus, dass bereits bestehende Implementierungen zum Umgang mit Permutationsgruppen und Labelled Branching, sowie ein Generator für diskrete Strukturen eingebunden werden konnten. Die in Tabelle 5.1 enthaltenen Dateien wurden hierzu aus den bisher unveröffentlichten Implementierungen zu MOLGEN 5.0 (Informationen zu den vorangegangenen Versionen finden sich in [GKLM98]) übernommen. Die abstrakte Klasse generator<T> wird in [Gug05] beschrieben.

Name	Verwendung
groups.cpp groups.h	Implementierungen zu Permutationsgruppen und Labelled Branchings.
gen.h	abstrakte Klasse generator<T> zur Implementierung des Backtrackings.
gen_old.h defs.cpp defs.h	Auflösen von Abhängigkeiten.

Tabelle 5.1.: Aus dem MOLGEN 5.0 System übernommene Dateien

In Tabelle 5.2 wird eine kurze Beschreibung der neu entstandenen Files gegeben. Eine

Name	Verwendung
gf.cpp gf.h	Implementierung eines endlichen Körpers, Vektoren und Matrizen über diesen und der Gruppe $\Big(\big(GL_k(q) \times GF(q)^{*n}\big) \rtimes \big(Gal(q) \times S_n\big)\Big)$.
node.cpp node.h	Die Knoten des Backtrackbaums.
partition.cpp partition.h	Geordnete und ungeordnete Mengenpartitionen.
codecan.cpp codecan.h	Implementierung des Backtrackbaums und des Kanonisierers.

Tabelle 5.2.: Die im Zuge der Diplomarbeit entstandenen C++ Dateien

ausführliche Dokumentation der Klassen erhält man durch Aufruf der Datei `index.html`, die sich auf der beigefügten CD im Verzeichnis `doc` befindet.

5.1. Bereitstellung der Daten

Zur Bereitstellung der Daten werden Textdateien durch die implementierten Klassen ausgelesen. Dazu sind die folgenden Vorgaben an das Format einzuhalten. Beispieldateien finden sich auf der beiliegenden CD im Verzeichnis `examples`.

5.1.1. Der zugrunde liegende Körper

Die Implementierung des Körpers $GF(q)$ mit $q = p^r$ wurde durch Nummerierung der Elemente von 0 bis $q - 1$ erreicht. Dabei wurde als einzige Voraussetzung festgelegt, dass die Null auch mit 0 und die Eins des Körpers mit 1 nummeriert sei. Die weiteren Elemente können beliebig gesetzt werden. Die gewählte Nummerierung der Elemente gibt gleichzeitig die Totalordnung des Körpers und beeinflusst somit die kanonische Form der Generatormatrix.

Das Format des Eingabefiles für den endlichen Körper muss von folgender Form sein:

- p r q
- Additionstabelle ($q \times q$)
- Multiplikationstabelle ($q \times q$)
- additive Inverse (q–Vektor)
- multiplikative Inverse (q–Vektor)
- Index eines primitiven Elements
- falls $r > 1$, die Bilder des Frobenius Automorphismus (q–Vektor)

5.1.2. Die Generatormatrix

Das Einlesen der Generatormatrix $\Gamma \in GF(q)^{k \times n}$ erfolgt zeilenweise, die Einträge müssen weiterhin aus $\{0, \ldots, q-1\}$ gewählt werden. Als weitere Argumente des Eingabefiles muss die Dimension der Matrix übergeben werden. Die Eingabe einer Nullmatrix führt im Programm zum Fehler. Wird eine Matrix mit nicht vollem Zeilenrang übergeben, so wird sie durch eine Generatormatrix des Codes ersetzt. Die Ausgabe bezieht sich auf die Generatormatrix, d.h. die Matrixkomponenten der Automorphismen und des Transporters haben eine niedrigere Dimension.

Das File muss die folgende Form einhalten:

- n k
- Γ (zeilenweise)

5.1.3. Bekannte Automorphismen

Zur Beschleunigung des Programms ist es weiterhin möglich, bereits bekannte Automorphismen zu übergeben. Da der Kanonisierungsalgorithmus ohnehin eine Verschmelzung von Spalten, welche Vielfache voneinander sind, vornimmt, ist es nicht nötig diese Permutationen zur Verfügung zu stellen. Die Permutationskomponenten der übergebenen Automorphismen werden anschließend in ein vollständiges Labelled Branching übergeführt und über dieses verwaltet. Für die Übergabe stehen zwei Varianten zur Verfügung:

Permutationskomponente Der Typ der Eingabe wird durch das Zeichen `p` zu Beginn der Datei kenntlich gemacht. Anschließend folgen die Permutationskomponenten der Automorphismen in Listen– oder Zykelschreibweise (eine Mischung ist erlaubt) durch Kommata getrennt. Die Listenschreibweise beginnt mit `[` und endet mit `]`, innerhalb der Klammern werden die Bilder der n Punkte durch Kommata getrennt aufgeführt. Bei der Zykelschreibweise werden die einzelnen Zykel durch $(\ldots)$ notiert.

Matrix + Körperautomorphismus Hier erfolgt die Eingabe der Automorphismen über den Anteil der operierenden $k \times k$–Matrix A, sowie einer Potenz t des Frobenius Automorphismus, welche dem Körperautomorphismus entspricht. Ausgehend von dieser Information wird eine Permutation berechnet, so dass die Generatormatrix durch entsprechende Multiplikation der Spalten auf sich selbst übergeführt werden kann. Die Eingabe erfolgt in der Form „`A, t`“, wobei A wiederum zeilenweise eingelesen wird. Weitere Automorphismen sind durch Kommata zu trennen. Die Kennzeichnung des Typs erfolgt durch die Eingabe `m` zu Beginn der Datei.

5.2. Das Programm

Um ein ausführbares Programm zu erhalten, wurde neben den oben genannten Dateien noch eine Datei `main.cpp` implementiert. Dadurch ist die Funktionalität über die Konsole erreichbar.

Die Ausgabe der Automorphismengruppe erfolgt ohne Angabe eines Erzeugers der trivialen Automorphismen

$$\mathcal{Z}_{n,k} := \left\{ \left((\mu^{-1} \cdot I_k, \{\mu\}^n); (\tau^0, \mathrm{id}_{S_n}) \right) \middle| \mu \in GF(q)^* \right\} \trianglelefteq Aut(\Gamma).$$

Die Gruppenordnung geben wir analog zu [BBF+06] zur Faktorgruppe

$$Aut(\Gamma)/\mathcal{Z}_{n,k}$$

an, d.h. die Ordnung ist um den Faktor $q-1$ reduziert.

5.2.1. Installation

Im Verzeichnis `src` befinden sich die genannten `C++` Dateien. Die Übersetzung erfolgt mittels des Aufruf von `make`. Hierbei muss bei älteren Compilerversionen die Option `-ffriend-injection` in der Datei `Makefile` entfernt werden. Diese Option ist ab `gcc` Version 4.1 verfügbar und notwendig um die Implementierungen aus `MOLGEN 5.0` zu übersetzen. Die Übersetzung der in dieser Arbeit implementierten Dateien erfolgt auch unter den Optionen `-pedantic` und `-Wall` ohne Fehler. Die auftretenden Fehlermeldungen beziehen sich allesamt auf die Dateien, welche übernommen wurden.

5.2.2. Kanonisierung

Mit dem Aufruf

`codecan` *Körper* `-s` *Generatormatrix [bekannte Automorphismen]*

kann man die kanonische Form einer Generatormatrix über einen endlichen Körper berechnen. Dabei müssen die übergebenen Files obigen Formatbedingungen genügen.

5.2.1 Beispiel. Wir rufen die Kanonisierung der Generatormatrix aus Beispiel 4.3.12:

```
codecan gf2.txt -s code7_2_4_2iso1.txt
```

bzw. mit einem File bekannter Automorphismen

```
codecan gf2.txt -s code7_2_4_2iso1.txt aut7_2_4_2iso1.txt
```

5.2.3. Bestimmung der Automorphismengruppe

Ist man nur an der Automorphismengruppe interessiert, so kann man dass Programm auch mit der Option „`-a`“ statt „`-s`“ starten. Die Laufzeitanalyse im Anhang A.1 ergibt jedoch im Durchschnitt eine schlechtere Laufzeit als der Kanonisierer.

5.2.2 Beispiel. Das selbe Beispiel wie oben, jedoch mit der Option zur Bestimmung der Automorphismengruppe:

```
codecan gf2.txt -a code7_2_4_2iso1.txt
```

bzw. mit einem File bekannter Automorphismen

```
codecan gf2.txt -a code7_2_4_2iso1.txt aut7_2_4_2iso1.txt
```

5.2.4. Der Äquivalenztest

Zum Test auf Äquivalenz dienen die Optionen „`-e`“ und „`-eno`“. Dabei wird mit der Option „`-eno`“ im Gegensatz zu „`-e`“ keine Kanonisierung der ersten Generatormatrix durchgeführt. Diese Variante stellt sich in der Laufzeitanalyse als ungünstiger dar. Wir benutzen aus diesem Grunde in den Beispielen nur die Option „`-e`“.

Wir benötigen als Eingabe noch eine weitere Generatormatrix, der Aufruf muss daher von folgender Gestalt sein:

`codecan` *Körper* `-e` *Generatormatrix1 Generatormatrix2 [bekannte Automorphismen1]*

Die Übergabe eines Automorphismenfiles ist ausschließlich für die erste Generatormatrix vorgesehen (bei „`-eno`“ entfällt diese Option). Nur diese wird kanonisiert und somit ist auch für diese die Kenntnis von Automorphismen von Vorteil, siehe hierzu auch Abschnitt 2.5.2.

Die Ausgabe beinhaltet bei positiver Beantwortung auch ein Gruppenelement, welches *Generatormatrix2* auf *Generatormatrix1* abbildet.

5.2.3 Beispiel. Wir testen ob die beiden Repräsentanten aus Beispiel 4.3.12 semilinear isometrische Codes erzeugen:

```
codecan gf2.txt -e code7_2_4_2iso1.txt code7_2_4_2iso4.txt
```

bzw. mit einem File bekannter Automorphismen

```
codecan gf2.txt -e code7_2_4_2iso1.txt code7_2_4_2iso4.txt
aut7_2_4_2iso1.txt
```

5.2.5. Weitere Optionen

Im Zuge der Implementierung und der Verifikation des Programms entstanden weitere Optionen zum Aufruf. Insbesondere erwies sich die Zufallserzeugung von Generatormatrizen und der anschließende Test der wichtigsten Klassenmethoden als sehr vorteilhaft. Desweiteren ist es auch möglich, die Kanonisierung mit linear abhängigen Spalten zu starten oder keine Rücktransformation der Ergebnisse vornehmen zu lassen. Die verfügbaren Optionen sind durch Aufruf der ausführbaren Datei mit Parameter „`-h`“ ersichtlich.

Als weitere nützliche Option erwies sich bei der Programmierung, das Backtracking über ein LaTeX–File zu dokumentieren. Die im Anhang dargestellten Bäume sind durch das Programm automatisch generiert. Um dieses Werkzeug zu aktivieren, ist es notwendig im Quellcode der Datei `codecan.cpp` die Konstante `LATEX` gleich 1 bzw. 2 zu setzen. In Stufe 1 erfolgt die Ausgabe ohne die i-minimalen Matrizen anzugeben, in Stufe 2 erfolgt diese Ausgabe. Durch die Konstante `LATEXOUT` wird der Dateiname festgelegt. Gegebenenfalls sind die Parameter der `pspicture` Umgebung von Hand anzupassen. Eine Beschreibung der Knotenbeschriftungen befindet sich in Anhang A.2.

6. Zusammenfassung und Ausblick

6.1. Fazit

In der Diplomarbeit wurde ein Algorithmus entwickelt, der in der Lage ist, eindeutige Repräsentanten der semilinearen Isometrieklassen linearer Codes zu berechnen. Das Verfahren nutzt die Beschreibung der Isometrieklassen als Bahnenmenge einer Gruppenoperation der symmetrischen Gruppe S_n und beruht auf dem Prinzip des Backtrackings. Um frühzeitig Teilbäume von der Suche ausschließen zu können, werden zum einen $S_n^{(i)}$-Homomorphismen zum Einsatz gebracht, andererseits die Gruppe der bekannten Automorphismen als vollständiges Labelled Branching verwaltet und nur Repräsentanten der Linksnebenklassen dieser Untergruppe untersucht. Somit erhält man als Ausgabe weiterhin auch ein Erzeugendensystem der Automorphismengruppe.

Auf Basis dieses Verfahrens entstanden weiterhin ein Äquivalenztest und ein gesonderter Algorithmus zur Bestimmung der Automorphismengruppe, der jedoch in seinem Laufzeitverhalten schlechter abschnitt als der Kanonisierer. Eine kurze Begründung zu diesem Verhalten wird im Anhang der Arbeit gegeben.

Der Algorithmus wurde in einer Form implementiert, so dass weitere Verfeinerungen gemäß Homomorphieprinzip eingefügt werden können. Hier gibt es sicherlich in den Bereichen der projektiven Geometrie und Codierungstheorie eine Vielzahl weiterer Möglichkeiten. Es ist zu erwarten, dass deren Effizienzen bei verschiedenen Codeklassen stark variieren können. Man vergleiche hierzu das Softwarepaket zur Kanonisierung von Graphen [McK], hier ist eine benutzerdefinierte Auswahl der Homomorphismen zur Effizienzsteigerung möglich. Will man diese Auswahl dem Benutzer abnehmen, so ist es nötig geeignete Heuristiken zur Verfügung zu stellen. Werden weitere Homomorphismen in das Programm eingefügt, so kommt man nicht umhin, sich dieses Sachverhalts näher zu widmen.

An dieser Stelle sei noch darauf hingewiesen, dass in der vorliegenden Form keine Homomorphismen angewandt werden, die zusätzlich noch von dem vorliegenden Stabilisator abhängen. Solche Homomorphismen können auf jeder Ebene iteriert angewandt werden, bis keine Änderung des Stabilisators[1] auftritt – das Prinzip der iterierten Verfeinerung. In diesem Zusammenhang würde sich vor allem das Punktieren und Verkürzen der Generatormatrix an Indexmengen, die durch Vereinigung von Blöcken hervorgehen, anbieten. Eine nähere Untersuchung in diesem Bereich wurde im Rahmen dieser Arbeit

[1] Dies drückt sich durch eine unveränderte Spalteneinteilung aus.

aber nicht durchgeführt. Es wurde jedoch ein allgemeines Verfahren angegeben, wie aus Codeinvarianten solche Homomorphismen gewonnen werden können.

Eine Fortentwicklung in diesem Bereich und der Einsatz entsprechender Heuristiken zur Auswahl und Reihenfolge der Homomorphismen würde sicherlich zu einer weiteren erheblichen Beschleunigung des Programms führen. Bislang führt ein Vergleich der Laufzeit mit dem in MAGMA [BCP97] implementierten Algorithmus von Leon [Leo82] zu erheblich längeren Laufzeiten. Jedoch ist dieser Vergleich nur bedingt aussagefähig. Der Vergleich ist nur mittels linearer Isometrie und über den Äquivalenztest bzw. der Bestimmung der Automorphismengruppe möglich. Die Laufzeitanalyse im Anhang zeigt aber gerade Defizite in diesem Bereich auf, da der in dieser Diplomarbeit entwickelte Algorithmus die Kanonisierung als Hauptziel verfolgt. Außerdem behandelt der Algorithmus von Leon nur lineare Isometrie und die Aufrufmöglichkeiten innerhalb von MAGMA sind auf Primkörper[2] und Ordnungen $q \leq 113$ beschränkt[3]. Insofern lässt sich hierbei nicht beobachten, inwiefern sich die unterschiedliche Wahl der operierenden Gruppe – bei Leon ist dies $S_{n\cdot(q-1)}$ – auswirkt.

Algorithmen zur Berechnung kanonischer Repräsentanten wurden bei der Recherche des Autors nur bedingt gefunden. Hier ist einzig das in [Ks06] beschriebene Verfahren, das den Code in einen Graphen transformiert und diesen kanonisiert, zu erwähnen. Jedoch ist die Ausgabe ein kanonischer Graph, eine Rücktransformation wird nicht angegeben und scheint auch nicht möglich. Das Vorgehen nutzt die Stärke des Programms nauty [McK] zur Kanonisierung von Graphen und wird vielmehr als Äquivalenztest vorgeschlagen. Auch hier ist eine direkte Abhängigkeit von der Ordnung des Körpers gegeben, da die Punkt-Hyperebenen-Beziehung der projektiven Geometrie $PG_{k-1}(q)$ genutzt wird. Der Graph besteht somit aus $2 \cdot \frac{q^k-1}{q-1}$ Punkten. Ob das Verfahren als $S_n^{(i)}$–Homomorphismus zum Einsatz in dem hier entwickelten Verfahren geeignet ist, konnte ebenfalls nicht untersucht werden.

6.2. Isometrieklassen linearer Codes über endlichen Kettenringen

In den 90er Jahren wurde entdeckt, dass der nichtlineare, binäre Kerdock Code K_2 als linearer Code über $\mathbb{Z}_4$ konstruiert werden kann – wir bezeichnen einen Untermodul C von ${}_RR^n$ als linearen Code der Länge n über R. Dieser Code hat die Parameter $(n, M, d) = (16, 2^8, 6)$[4], die von keinem linearen Code über $GF(2)$ erreicht werden können. Der Übergang von $\mathbb{Z}_4$ nach $GF(2)$ erfolgt dabei mittels der Gray-Abbildung, einer Isometrie $(\mathbb{Z}_4, w_{Lee}) \to (GF(2), w_{Ham})$, wobei w_{Ham} das Hamminggewicht und w_{Lee} das Lee-Gewicht bezeichne.

[2] mit Ausnahme von $GF(4)$

[3] im Paket GUAVA [BBC+07] des Computeralgebrasystems GAP [GAP06] sogar auf $q = 2$

[4] mit $M := |C|$

In der Folge stellte man fest, dass noch viele weitere gute Blockcodes über Körpern als lineare Codes über Ringen konstruiert werden können.

Ein endlicher, assoziativer Ring mit Eins heißt Kettenring, falls der Linksidealverband eine Kette bildet. Die Verallgemeinerung auf diese Ringklasse bietet sich aus folgenden Gründen heraus an:

- Die algebraischen Eigenschaften liegen nahestmöglich an denen der Körper, Begriffe wie Generatormatrix, dualer Code oder Gewicht lassen sich geeignet verallgemeinern.
- Sie umfasst die Galoisringe $GR(p^{mr}, p^m)$ und somit insbesondere $GF(q)$ und $\mathbb{Z}_q$. Dies sind wichtige Spezialfälle in denen bereits vielversprechende Erfolge erzielt wurden.
- Die linearen Codes über einen Kettenring R können auch als Multimengen von Punkten einer projektiven Hjelmslev-Geometrie über R gesehen werden. Dies ist eine naheliegende Verallgemeinerung des wohlbekannten Zusammenhangs der linearen Codes über Körpern und der klassischen projektiven Geometrie $PG_{k-1}(q)$.

Als Gewichtsfunktion in R wählt man das sogenannte homogene Gewicht w_{hom} mit der folgenden Definition:

$$w_{hom} : R \to \mathbb{R}$$

$$w_{hom}(a) := \begin{cases} 0; & a = 0 \\ q; & a \in N^{m-1} \setminus \{0\} \\ q-1; & \text{sonst} \end{cases}$$

wobei N das maximale Ideal von R, q die eindeutige Primzahlpotenz mit $R/N \simeq GF(q)$ und $m =$ (Anzahl der Ideale in R)-1 bezeichne. Für einen Vektor $c \in R^n$ setzt man analog zum Hamming-Gewicht $w_{hom}(c) = \sum_{i \in n} w_{hom}(c_i)$. Eine geeignete Verallgemeinerung der Gray-Abbildung $(R, d_{hom}) \to (GF(q)^{q^{m-1}}, q^{2-m} d_{Ham})$ führt den R-linearen Code C in einen Blockcode über $GF(q)$ mit gleicher Gewichtsverteilung über.

Wir bezeichnen zwei R-lineare Codes der Länge n wie in der klassischen Codierungstheorie als äquivalent, falls eine Isometrie $\iota : (R^n, w_{hom}) \to (R^n, w_{hom})$ existiert, die den einen auf den anderen überführt. Aus codierungstheoretischer Sicht ist man weiterhin nur an Repräsentanten dieser Äquivalenzklassen interessiert.

Ein analoges Vorgehen zur Bestimmung eindeutiger Repräsentanten gemäß des in der Diplomarbeit entwickelten Algorithmus bei Codes über Kettenringen scheint möglich. Der lineare Code C über dem Ring R lässt sich analog durch Angabe einer Matrix formulieren. Die Zeilen dieser Matrix werden von Codewörtern gebildet, die den Untermodul erzeugen. Zwei $k \times n$-Matrizen Γ_1, Γ_2 erzeugen den gleichen Code, falls es eine invertierbare Matrix $A \in GL(k, R)$ gibt mit $A\Gamma_1 = \Gamma_2$.

Die Permutation der Koordinaten, beliebige Multiplikation der Einträge einer Koordinate mit Einheiten R^* und die komponentenweise Anwendung von Ringautomorphismen $Aut(R)$ erhalten, wie im Falle der linearen Codes über Körpern, weiterhin die Gewichte. Der Begriff der semilinearen Isometrie lässt sich auf Ringe übertragen. Zwei Codes $C_1, C_2 \leq {}_R R^n$ sind semilinear isometrisch, falls

$$\exists \alpha \in Aut(R), \varphi \in (R^*)^n, \pi \in S_n : (\varphi; (\alpha, \pi))C_1 = C_2.$$

Nehmen wir die Operation der invertierbaren Matrizen hinzu, so können wir die Isometrieklassen linearer Codes über endlichen Kettenringen nach Kapitel 3 als Bahnen der Operation eines semidirekten Produkts auf den Generatormatrizen formulieren:

$$\big(GL(k,R) \times R^{*n}\big) \rtimes \big(Aut(R) \times S_n\big) \backslash\backslash R^{k \times n}$$

Diese Operation können wir durch die gleichen Überlegungen wiederum durch die äquivalente Operation der symmetrischen Gruppe S_n auf einer Bahnenmenge formulieren:

$$S_n \backslash\backslash \Big[\Big(\big(GL(k,R) \times R^{*n}\big) \rtimes Aut(R) \Big) \backslash\backslash R^{k \times n} \Big]$$

Für den Kanonisierungsalgorithmus über Kettenringen bleibt nun zu prüfen, ob sich die Berechnung von kanonischen Repräsentanten für die innere Operation weiterhin so einfach darstellt, wie im Falle der Kanonisierung über Körpern, siehe Abschnitt 4.1. Falls dies nicht der Fall ist, muss eine geeignete Prozedur bereitgestellt werden. Weiterhin muss geprüft werden, ob sich die angewandten $S_n^{(i)}$-Homomorphismen übertragen lassen.

A. Anhang

A.1. Laufzeiten

An dieser Stelle soll nun das Laufzeitverhalten des Programms näher untersucht werden. Wir betrachten hierzu jeweils die durchschnittliche Laufzeit über 500 zufällig erzeugte Generatormatrizen mit Parametern (n, k, q) und vollem Zeilenrang k. Allein die Permutation der Koordinaten führt bei einzelnen Beispielen zu erheblichen Laufzeitunterschieden, da das Abschneiden gemäß Homomorphieprinzip allein von der Güte des bislang optimalen Kandidaten abhängig ist, siehe Abschnitt A.3. Zum Beispiel wurde bei der Kanonisierung, der semilinear isometrischen $(20, 10, 3)$ – Codes `code20_10_3min.txt` und `code20_10_3max.txt`, Laufzeiten zwischen 0.313 und 6.907 Sekunden gemessen. Die maximale gemessene Laufzeit für die Kanonisierung eines Codes mit diesem Parametersatz war 68.609 Sekunden.

Zunächst halten wir die Länge $n = 20$ der Codevektoren konstant. Es ergibt sich die Tabelle A.1. Bei $k > \lfloor \frac{n}{2} \rfloor$ wechselt das Programm von den Generatormatrizen zu den Kontrollmatrizen mit Parameter $(n, n - k)$, aus diesem Grunde ist die Zeile $k = 10$ gesondert markiert. Die Tabelle A.1 ist symmetrisch zu dieser Zeile, der Aufwand beim Übergang zur Kontrollmatrix ist im Vergleich zur Gesamtlaufzeit nahezu vernachlässigbar. Die geringfügigen Unterschiede ergeben sich vielmehr aus stochastischen Gründen.

Der Übergang von k zu $k + 1$ wirkt sich durch eine Vervielfachung der Laufzeit aus. Jedoch ist hier weder eine direkte Abhängigkeit von q noch von k ersichtlich.

Tabelle A.2 zeigt im Gegensatz dazu das Verhalten für eine feste Dimension $k = 5$ der linearen Codes. Hier ist im Gegensatz zu Tabelle A.1 nur ein geringfügiger Anstieg des Aufwands zu verzeichnen. Dies liegt daran, dass durch die Festlegung der k Einheitsvektoren in den Spalten eines i-minimalen Repräsentanten die Homomorphismen $f_{i,1}$ und $f_{i,2}$ bereits starke Einschränkungen liefern. Sobald nun für das Stabilisatortripel die Zeilenpartition einelementig ist und der Körperautomorphismus nur noch aus der Identitität besteht, liefert der Homomorphismus $f_{i,3}$ eine Partition der Spalten in einelementige Blöcke. Dies ist auch die Begründung für das stärkere Wachstum in Abhängig von k.

Für die Spalte $q = 2$ sind die Zeiten mehr oder minder konstant, da sich hier bereits die Verschmelzung von identischen Spalten[1] auswirkt. Die Anzahl der Punkte in der

[1] Spalten die Vielfache voneinander sind müssen im binären Fall identisch sein

projektiven Geometrie $PG_4(2)$ ist $2^5 - 1 = 31$. Somit ist bei 20 zufällig ausgewählten Spalten durchaus zu erwarten, dass einzelne Vektoren mehrfach ausgewählt wurden.

Die Erläuterungen zu den Tabellen A.3 und A.4 der Algorithmen gemäß Abschnitt 2.5 finden sich in Anhang A.3. Die geringeren Laufzeiten für $q = 2, 3, 4, 5$ der ersten Zeile im Gegensatz zu Tabelle A.2 ergeben sich durch die Tests gemäß der Homomorphismen auf Ebene 0, Abschnitt 4.3.1, welche bereits vor der eigentlichen Kanonisierung durchgeführt werden.

k	$q=2$	$q=3$	$q=4$	$q=5$	$q=8$	$q=16$	$q=1009$
3	0,00041	0,00047	0,00078	0,00091	0,00384	0,00997	1,20431
4	0,00059	0,00172	0,00672	0,01050	0,02044	0,02600	1,65753
5	0,00216	0,01534	0,02903	0,03603	0,06953	0,13728	3,95544
6	0,01153	0,05559	0,07966	0,12169	0,18294	1,01378	62,18010
7	0,04787	0,14919	0,21584	0,34022	1,17313	4,64700	
8	0,15925	0,35384	0,75916	1,57797	6,56325	28,45380	
9	0,39591	1,12181	3,12003	8,98147	35,60480		
10	0,75172	3,49434	16,23130	38,98710			
11	0,40828	1,20232	3,05500	6,34753			
12	0,16159	0,39294	0,75291	1,41450			
13	0,05553	0,15516	0,21994	0,32803			
14	0,01322	0,05697	0,09128	0,12165			
15	0,00187	0,01606	0,03159	0,03497			
16	0,00075	0,00253	0,00709	0,01050			
17	0,00044	0,00090	0,00110	0,00106			

Tabelle A.1.: Laufzeit in Sekunden für $n = 20$

n	$q=2$	$q=3$	$q=4$	$q=5$	$q=8$	$q=16$	$q=1009$
10	0,00178	0,00538	0,00944	0,01103	0,02119	0,05581	3,03525
11	0,00200	0,00694	0,01153	0,01372	0,02019	0,06887	5,95722
12	0,00209	0,00775	0,01378	0,01709	0,02403	0,05928	9,47453
13	0,00259	0,00916	0,01500	0,01931	0,03094	0,05706	12,38210
14	0,00209	0,01016	0,01559	0,02178	0,03850	0,05809	10,60960
15	0,00309	0,01078	0,01681	0,02187	0,04522	0,06385	7,72778
16	0,00191	0,01131	0,01900	0,02472	0,05244	0,07066	4,07188
17	0,00272	0,01175	0,02100	0,02709	0,05828	0,08519	2,36541
18	0,00178	0,01369	0,02422	0,03000	0,06825	0,09669	2,80503
19	0,00222	0,01506	0,02597	0,03191	0,06884	0,12094	3,35431
20	0,00216	0,01534	0,02903	0,03603	0,06953	0,13728	4,17147

Tabelle A.2.: Laufzeit in Sekunden für den Fall $k = 5$

n	$q=2$	$q=3$	$q=4$	$q=5$	$q=8$	$q=16$
10	0,00241	0,01912	0,04309	0,06934	0,12794	0,33766
11	0,00275	0,02806	0,06441	0,10034	0,20403	0,46062
12	0,00269	0,03391	0,08444	0,14169	0,29322	0,74669
13	0,00244	0,04025	0,11422	0,18844	0,43838	1,04266
14	0,00212	0,04334	0,14128	0,24331	0,62019	1,49175
15	0,00184	0,04947	0,16609	0,31522	0,80213	2,09997
16	0,00166	0,04800	0,19138	0,39531	1,11506	2,84741
17	0,00141	0,05631	0,22825	0,46416	1,33244	3,68587
18	0,00144	0,04969	0,27287	0,51453	1,66166	4,95816
19	0,00106	0,05025	0,27134	0,65497	2,01819	6,10688
20	0,00103	0,04969	0,31003	0,68950	2,38656	7,82272

Tabelle A.3.: Laufzeit des Automorphismengruppenalgorithmus nach Abschnitt 2.5 in Sekunden für den Fall $k = 5$

Γ_0 kanonisiert	$q=2$	$q=3$	$q=4$	$q=5$	$q=8$	$q=16$
ja (Zeit inkl. Kanonisierung)	0,00028	0,00394	0,01603	0,02833	0,07108	0,18890
nein	0,00035	4,59315	4,73987	5,12091	6,89020	7,59379

Tabelle A.4.: Laufzeiten des Äquivalenztests nach Abschnitt 2.5 in Sekunden für den Fall $n = 20, k = 5$ mit und ohne Kanonisierung der ersten Generatormatrix

A.2. Die generierten Backtrackbäume des Programms

An dieser Stelle sei kurz beschrieben, wie die durch das Programm automatisch generierten Backtrackbäume, vgl. Abschnitt 5.2.5, zu lesen sind:

- Der Baum wird in Tiefensuche von links nach rechts abgearbeitet.
- Die Knoten des Backtrackbaums stellen unter `LATEX` $= 2$ den aktuellen Knoten als i-minimalen Repräsentanten dar, die Blöcke der kanonischen Younguntergruppen seien wie in der Diplomarbeit durch die Unterteilung der Spalten der Matrix gegeben.
- Die Kanten des Baums sind mit dem Rechtstransversalenelement beschriftet. Die aktuelle Permutation g_i ergibt sich also durch Multiplikation der Beschriftungen an dem Pfad zur Wurzel.
- Der Buchstabe `N` symbolisiert, dass ein neuer optimaler Kandidat gefunden wurde.
- Der Buchstabe `A` steht für einen Automorphismus. Wir haben ein Blatt erreicht, das zu einem Blatt identisch ist, welches mit `N` gekennzeichnet ist und am weitesten rechts steht. Wir erhalten den Automorphismus durch $g_{opt}^{-1} g_i$.
- An Knoten, welche mit `C` beschriftet sind, ergab sich eine Spalte i, die größer ist als beim optimalen Kandidaten (Verletzung der i-Minimalität).
- Ein Knoten, der mit `L` beschriftet ist, wurde wegen des Tests mittels des vollständigen Labelled Branchings zurückgewiesen.
- Die weiteren Beschriftungen von Knoten stehen für das Abschneiden von Teilbäumen nach dem Homomorphieprinzip. Aus technischen Gründen beziehen sie sich aber nicht auf die aktuelle Ebene sondern auf den Vaterknoten. Als Beschriftungen treten auf
  - `R`: Homomorphismus $f_{i,1}$ (Rank-Refine)
  - `S`: Homomorphismus $f_{i,2}$ (Supp-Refine)
  - `D`: Homomorphismus $f_{i,3}$ (Division-Refine)

A.3. Ein Gegenbeispiel zu Abschnitt 2.5

Wir möchten nun aufzeigen, wie die oben aufgeführten Laufzeitunterschiede bei der Berechnung der Automorphismengruppe zustande kommen. Wir lassen uns hierzu von dem Programm `codecan` die Backtrackbäume zu der Generatormatrix

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

über $GF(4)$[2] bei der Berechnung der Automorphismengruppe mittels des Kanonisierers, sowie des speziellen Algorithmus gemäß Abschnitt 2.5.1, erzeugen. Ein Eingabefile `code6_3_3_4iso4.txt` befindet sich in dem Ordner `examples` auf der beigefügten CD. Die Ausgaben lassen sich aufgrund ihrer Abmessungen hier nur in vereinfachter Form wieder geben[3], siehe Abbildungen A.1 und A.2. Für eine ausführliche Dokumentation sei auf das elektronische Medium verwiesen: Die Dateien

- `backtracktree_can6_3_3_4iso4.ps`
- `backtracktree_aut6_3_3_4iso4.ps`

zeigen den Backtrackbaum des Kanonisierers und des weiteren Algorithmus. Die Argumentation erfolgt anhand dieser Dateien.

Nun zu der Begründung des Laufzeitverhaltens. Bei dem spezialisierten Algorithmus wird das zuerst erreichte Blatt als Referenz gewählt, das heißt bei Anwendung des Homomorphieprinzips wird stets mit diesem verglichen. Bei der Kanonisierung wird jedoch ein minimaler Kandidat herangezogen. Es wurden nun Ordnungen im Bildbereich gewählt, so dass unwahrscheinliche Konstellationen bevorzugt werden. Zum Beispiel werden Vektoren, welche im Erzeugnis der festgelegten Spalten liegen, innerhalb eines Blocks nach vorne sortiert. Die Wahrscheinlichkeit, dass drei zufällig ausgewählten Vektoren einen 2-dimensionalen Unterraum aufspannen ist geringer, als dass sie linear unabhängig sind.

So ergibt sich zwar im Backtrackbaum, dass zunächst auch der zweite Knoten auf Ebene 2 bearbeitet werden muss, jedoch erreichen wir hierdurch über den Pfad

id(45)(34)(23)(12)id

ein Blatt der Form

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

N

Im Folgenden werden also bei der Kanonisierung nur noch Knoten bearbeitet, welche linear abhängige Vektoren an den ersten 3 Positionen stehen haben. Dies ergibt für die restlichen Fälle eine starke Einschränkung. Bereits für den nächsten Teilbaum auf Ebene 1 mit Wurzel

$$\left(\begin{array}{c|ccccc} 1 & 0 & 0 & 0 & 2 & 3 \\ 0 & 1 & 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

[2] Arithmetik laut `gf4.txt`

[3] erreicht über die Option `LATEX` = 1

zahlt sich dies aus. Im Gegensatz zu dem Automorphismengruppenalgorithmus können wir bei der Kanonisierung bereits jeweils auf Ebene 2 die Untersuchungen abbrechen. Bei dem anderen Vorgehen müssen wir bis auf Ebene 3 vordringen, um dies zu erkennen. Wir haben aus diesem Grunde 16 Knoten mehr zu untersuchen.

Für den Äquivalenztest zu Γ_0, Γ_1, ergibt sich ein ähnliches Bild. Ziehen wir einen beliebigen Blattknoten $g\Gamma_0$ aus dem Backtrackbaum zu Γ_0 heran, so ist zu erwarten, dass die Bilder unter den Homomorphismen häufiger angenommen werden. Somit werden die Teilbäume später abgeschnitten. Dieser Mehraufwand kompensiert den Aufwand, der bei der zusätzlichen Kanonisierung von Γ_0 entsteht. Als weiterer Nachteil des Verfahrens nach Algorithmus 2.9 erweist sich noch, dass hier die Automorphismengruppe nicht berücksichtigt werden kann.

Als Resultat dieser Untersuchungen stellen wir fest, dass wir, um schnelle Algorithmen für diese beiden Fälle erreichen zu können, eine „gute“ Vorsortierung der Spalten vornehmen müssen. Das heißt, es sollte ein Referenzknoten gewählt werden, dessen Bilder unter den einzelnen Homomorphismen möglichst mit geringer Wahrscheinlichkeit angenommen werden. Eine entsprechende Heuristik konnte im Rahmen der Diplomarbeit nicht implementiert werden.

Weiterhin halten wir fest, dass die Laufzeitunterschiede auf eine geeignete Auswahl der Ordnungen in den Bildbereichen der Homomorphismen für den Kanonisierungsalgorithmus hindeuten.

Abbildung A.1.: Gegenbeispiel: Kanonisierer

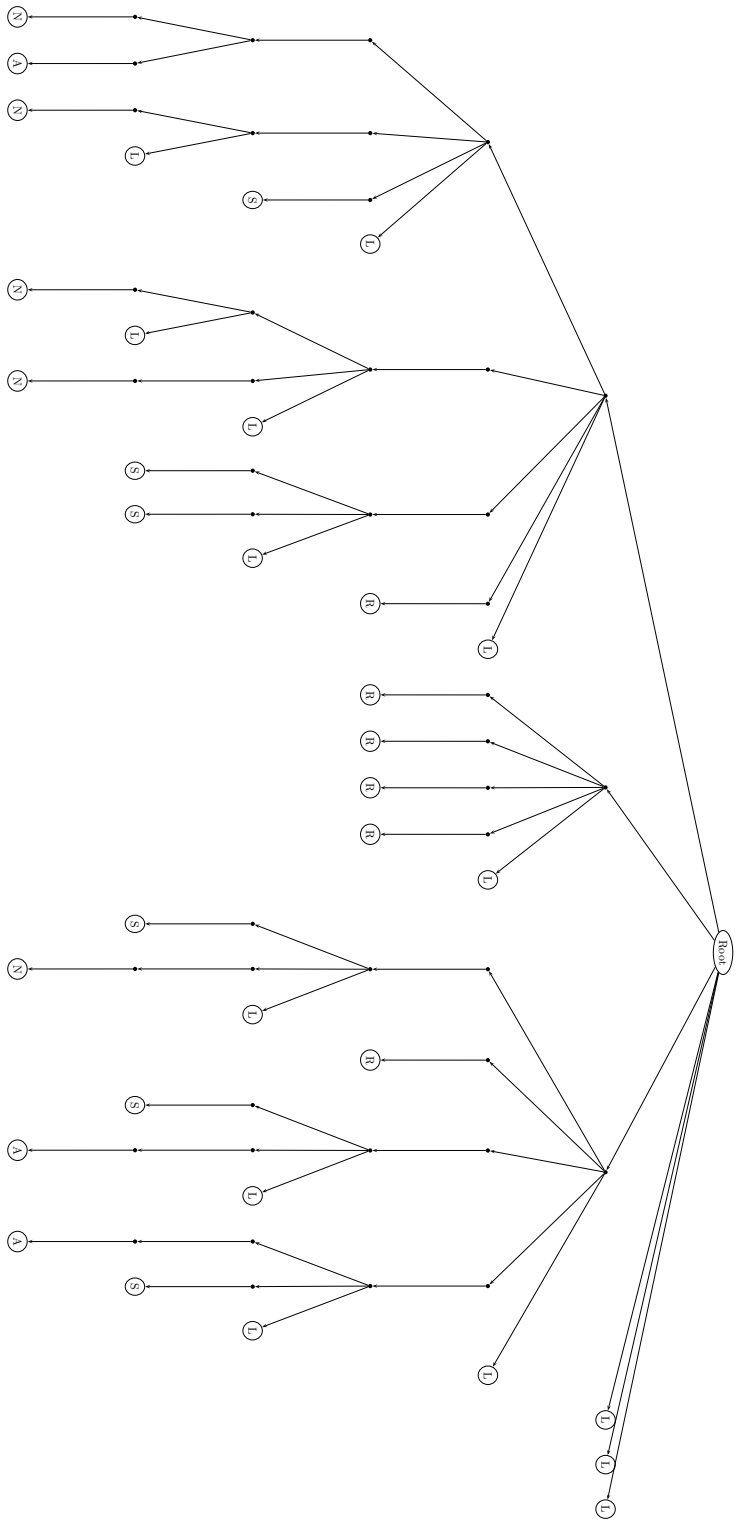
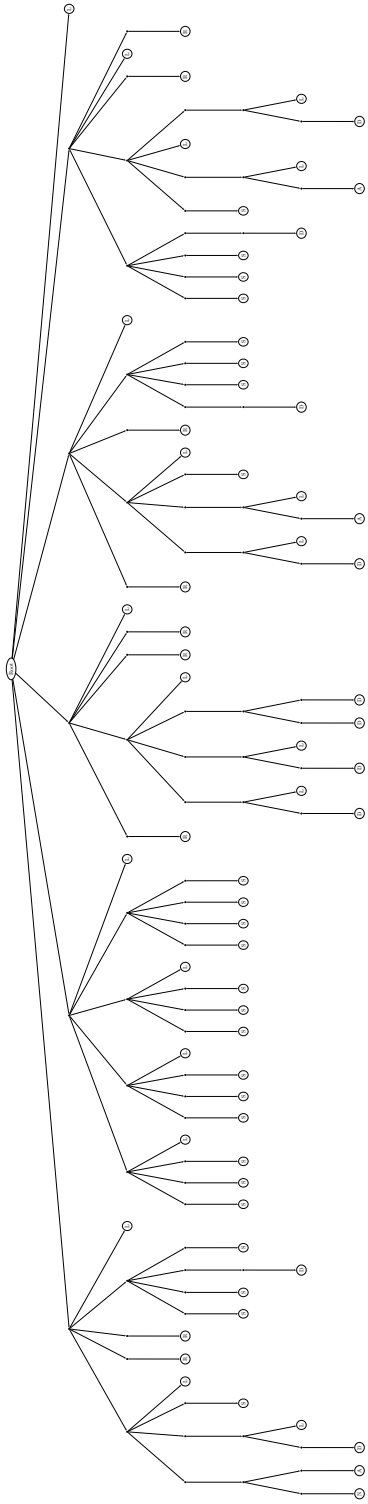


Abbildung A.2.: Gegenbeispiel: Automorphismengruppenalgorithmus



Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[BBC+07] Baart, Reinald ; Boothby, Tom ; Cramwinckel, Jasper ; Joyner, David ; Miller, Robert ; Minkes, Eric ; Roijackers, Erik ; Ruscio, Lea ; Tjhai, Cen: *A GAP package for computing with error-correcting codes, Version 3.0.* August 2007. – `http://www.gap-system.org/Packages/guava.html`

[BBF+06] Betten, Anton ; Braun, Michael ; Fripertinger, Harald ; Kerber, Adalbert ; Kohnert, Axel ; Wassermann, Alfred: *Error Correcting-Linear Codes: Classification by Isometry and Applications.* Berlin, Heidelberg : Springer-Verlag, 2006

[BCP97] Bosma, Wieb ; Cannon, John ; Playoust, Catherine: The Magma algebra system. I. The user language. In: *J. Symbolic Comput.* 24(3-4) (1997), S. 235–265. – `http://magma.maths.usyd.edu.au/magma/`, Magma Version 2.14

[GAP06] The GAP Group (Hrsg.): *GAP – Groups, Algorithms, and Programming, Version 4.4.9.* The GAP Group, 2006. – `http://www.gap-system.org/gap.html`

[GKLM98] Grüner, Thomas ; Kerber, Adalbert ; Laue, Reinhard ; Meringer, Markus: MOLGEN 4.0. In: *MATCH* 37 (1998), März, S. 205–208. – `http://www.molgen.de/`

[Gug05] Gugisch, Ralf: *Konstruktion von Isomorphieklassen orientierter Matroide.* Bayreuth : Bayreuther Mathematische Schriften, Heft 72, 2005

[Jer86] Jerrum, M.: A compact representation for permutation groups. In: *J. Algorithms* 7 (1986), S. 60–78

[Ker04] Kerber, Adalbert: *Vorlesungsskript Lineare Algebra und Algebra.* Universität Bayreuth, WS 2002/2003 - SS 2004

[Ks06] Kaski, Petteri ; Östergård, Patric R. J.: *Classification algorithms for codes and designs.* Berlin : Springer-Verlag, 2006

[Leo82] Leon, Jeffrey S.: Computing automorphism groups of error-correcting codes. In: *IEEE Transactions on Information Theory* 28 (1982), Mai, Nr. 3, S. 496–511

[Leo84] Leon, Jeffrey S.: Computing automorphism groups of combinatorial objects. In: *Computational Group Theory* (1984), S. 321–335

[McK] McKay, Brendan D.: *nauty User's Guide (Version 2.2).* Computer Science Department, Australian National University, ACT 0200, Australia. – `http://cs.anu.edu.au/~bdm/nauty/`

[McK81] McKay, Brendan D.: Practical Graph Isomorphism. In: *Congressus Numerantium* 30 (1981), S. 45–87. – `http://cs.anu.edu.au/~bdm/nauty/PGI/`

[Mon87] Montpetit, A.: Note sur la notion d'équivalence entre deux codes linéaires. In: *Discrete Mathematics* 65 (1987), S. 177–185

[Sen00] Sendrier, Nicolas: Finding the permutation between equivalent linear Codes: The Support Splitting Algorithm. In: *IEEE Transactions on Information Theory* 46 (2000), Juli, Nr. 4, S. 1193–1203

Erklärung

Hiermit erkläre ich, dass ich die Diplomarbeit selbst und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch in keinem anderen Prüfungsverfahren vorgelegt.

Wolfsgraben, den 3. Januar 2008