Extremale Gitter mit großen Automorphismen

Masterarbeit

von Simon Berger

Vorgelegt am Lehrstuhl D für Mathematik der RWTH-Aachen University

bei Prof. Dr. Gabriele Nebe (Erstgutachterin)
und Prof. Dr. Markus Kirschmer (Zweitgutachter)

16. September 2018

Inhaltsverzeichnis

1 Einleitung

Die Gittertheorie ist seit vielen Jahren ein wesentlicher Bestandteil der theoretischen Mathematik in Themengebieten wie der algebraischen Zahlentheorie und der Gruppentheorie. In der Anwendung sind oftmals Gitter von Interesse, welche besonders dichte Kugelpackungen liefern. Also Gitter, die im Vergleich zu ihrer Determinante ein möglichst großes Minimum besitzen. Die Klassifikation möglichst dichter Gitter stellt jedoch eine große Herausforderung dar.

H.-G. Quebbemann definiert in seiner Arbeit [Que95] den Begriff eines *modularen* Gitters und zeigt, dass die Thetareihen solcher modularen Gitter Modulformen einer bestimmten Gruppe sind. Diese Struktur erlaubt die Beschreibung sogenannter *extremaler* modularer Gitter, welche innerhalb der Klasse der modularen Gitter eine maximale Dichte haben. Die Klassifikation extremaler Gitter ist eines der großen Forschungsgebiete aus der Gittertheorie. In dieser Arbeit wird versucht, extremale Gitter mithilfe ihrer Automorphismengruppen zu untersuchen und zu konstruieren.

Zunächst definieren wir dazu unsere grundlegenden Begriffe und werfen einen genaueren Blick auf die Dichte extremaler Gitter. Anschließend beschreiben wir modulare Gitter L mit einer Struktur als gebrochene Ideale eines zyklotomischen Zahlkörpers $\mathbb{Q}(\zeta_m)$, sogenannte *Ideal-Gitter*. Diese Struktur ist gegeben, falls L einen Automorphismus σ mit $\mu_\sigma = \Phi_m$ besitzt, sodass $\varphi(m) = \mathrm{Dim}(L)$ gilt. Mithilfe dieser zusätzlichen Struktur lässt sich ein Algorithmus entwickeln, welcher zu gegebener Stufe, Determinante und Dimension eine vollständige Liste der Ideal-Gitter mit den festgelegten Parametern konstruiert. Dieser Algorithmus wurde von Michael Jürgens in [Jü15] beschrieben. Die einzelnen Schritte werden im Zuge dieser Arbeit genau untersucht und im Computer-Algebra-System `MAGMA` implementiert.

Im sich daran anschließenden Hauptteil der Arbeit erfolgt die Beschreibung extremaler modularer Gitter mit *großem* Automorphismus. Hier gehen wir lediglich von den Voraussetzungen $|\sigma| = m$ und $\Phi_m|\mu_\sigma$ mit $\varphi(m) > \frac{\mathrm{Dim}(L)}{2}$ aus. In diesem Falle induziert σ ein Teilgitter $M := L \cap \mathrm{Kern}(\Phi_m(\sigma)) \perp L \cap \mathrm{Kern}(\frac{\mu_\sigma}{\Phi_m}(\sigma))$. DIe erste Komponente hat dabei eine Struktur als Ideal-Gitter und kann deshalb mit dem Algorithmus des vorherigen Kapitels leicht konstruiert werden; für die zweite Komponente benötigen wir weitere Theorie. Jürgens und Nebe untersuchen in [Jü15] und [Neb13] Gitterautomorphismen σ mit Primzahlordnung p. Wie vorher induziert solch ein σ ein Teilgitter $M := L \cap \mathrm{Kern}(\Phi_p(\sigma)) \perp L \cap \mathrm{Kern}(\Phi_1(\sigma))$, in diesem Falle lassen sich allerdings wichtige Einschränkungen an die Determinanten der Komponenten sowie an den Index von M in L zeigen. Die Informationen über Automorphismen von Primzahlordnung können anschließend verwendet werden, um mittels der Primteiler von m auch Automorphismen allgemeinerer Ordnung zu untersuchen. Nebe wendet in [Neb13] diese Erkenntnisse an, um extremale unimodulare Gitter der Dimension 48 mit großem Automorphismus zu klassifizieren. In dieser Arbeit wird die dortige Vorgehensweise auf ℓ-modulare Gitter verallgemeinert und als Algorithmus zusammengefasst. Alle Voraussetzungen und Teilschritte werden dabei genau erläutert und in `MAGMA` implementiert. Auf diese Weise konnten insgesamt 33 bisher unbekannte extremale modulare Gitter konstruiert werden. Abschließend wird die Vollständigkeit der Ergebnisse genauer erzläutert. Dazu zählen wir alle Möglichkeiten für die charakteristischen Polynome von Gittern auf, die nicht die Voraussetzungen des Algorithmus erfüllen und somit gegebenenfalls nicht gefunden werden konnten.

2 Grundbegriffe

§ 2.1 Bilineare Vektorräume

Wir wiederholen zunächst einige wichtige Begriffe aus der Gittertheorie, welche wir in der Arbeit häufig benötigen werden. Zunächst führen wir das Konzept eines bilinearen Vektorraumes ein. Die nun angeführten Definitionen sind [Kne02, Def. (2.1)] entnommen.

(2.1.1) Definition

(i) Sei A ein Ring und E ein A-Modul. Für eine symmetrische Bilinearform $b : E \times E \to A$ heißt das Paar (E, b) ein *bilinearer A-Modul* (bzw. falls A ein Körper ist ein *bilinearer A-Vektorraum*).

(ii) Eine *isometrische Abbildung* (oder kurz *Isometrie*) zwischen zwei bilinearen Moduln (E, b) und (E', b') ist ein Modulisomorphismus $f : E \to E'$ mit $b(x, y) = b'(f(x), f(y))$.

(iii) Die Gruppe $O(E, b) := \{f : E \to E \mid \text{ f ist Isometrie}\}$ aller Isometrien eines bilinearen Vektorraums (E, b) in sich selbst heißt die *Isometriegruppe* von (E, b).

Nun folgen Definitionen zum Gitterbegriff, zu finden in [Kne02, Def. (14.1), (14.2)].

(2.1.2) Definition

(i) Sei K ein Körper, V ein endlich-dimensionaler K-Vektorraum mit Basis $(b_1, \ldots, b_n)$. Ein R-Gitter in V ist ein R-Untermodul L von V, zu dem Elemente $a, b \in K^*$ existieren mit $a \sum_{i=1}^{n} Rb_i \subseteq L \subseteq b \sum_{i=1}^{n} Rb_i$.

(ii) Sei b eine nicht-ausgeartete symmetrische Bilinearform auf V und L ein Gitter in V. Dann ist auch $L^{\#} := \{x \in V \mid b(x, y) \in R \text{ für alle } y \in L\}$ ein R-Gitter und heißt *das zu* L *duale Gitter* (bzgl. b).

(iii) Für $m \in \mathbb{N}$ heißt das Gitter $L^{\#,m} := \frac{1}{m}L \cap L^{\#}$ *partielles Dualgitter* von L.

(iv) Sei (V, b) ein bilinearer K-Vektorraum und L ein Gitter in V. Die Gruppe $\mathrm{Aut}(L) := \{\sigma : V \to V \mid \sigma \text{ ist Isometrie und } \sigma(L) = L\}$ heißt die *Automorphismengruppe* von L.

(2.1.3) Bemerkung

Falls R ein Hauptidealbereich ist, vereinfacht sich die Definition erheblich, da Teilmoduln von endlich erzeugten freien Moduln über Hauptidealbereichen wieder frei sind. Ein R-Gitter ist per Definition zwischen zwei freien Moduln eingespannt, also sind die R-Gitter in diesem Fall genau die freien R-Moduln von Rang n.

Insbesondere interessieren uns $\mathbb{Z}$-Gitter in $\mathbb{R}^n$. Für diese folgen nun ein paar weitere aus [Kne02, Def. (1.7), (1.13), (14.7), (26.1)] abgeleitete Definitionen.

(2.1.4) Definition

Sei L ein $\mathbb{Z}$-Gitter mit Basis $B = (e_1, \ldots, e_n)$ in $(\mathbb{R}^n, b)$, für eine symmetrische Bilinearform b.

(i) Die Matrix $G := \mathrm{Gram}(B) = (b(e_i, e_j))_{i,j=1}^n$ heißt *Gram-Matrix* von L, $\mathrm{Det}(L) := \mathrm{Det}(G)$ heißt die *Determinante* von L.

(ii) Das Gitter L heißt *ganz*, falls $b(L, L) \subseteq \mathbb{Z}$ gilt.

(iii) Das Gitter L heißt *gerade*, falls $b(x, x) \in 2\mathbb{Z}$ für alle $x \in L$ gilt.

(iv) Die *Stufe* von L ist die kleinste Zahl $\ell \in \mathbb{N}$, sodass $\sqrt{\ell}L^{\#}$ ein gerades Gitter ist.

(v) Das *Minimum* von L ist definiert als $\mathrm{Min}(L) := \min\{b(x, x) \mid 0 \neq x \in L\}$.

(2.1.5) Bemerkung

(i) Nach [Kne02, Satz (14.7)] gilt $\mathrm{Det}(L) = |L^{\#}/L|$. Insbesondere ist die Determinante für $\mathbb{Z}$-Gitter unabhängig von der Wahl der Basis. Allgemeiner ist die Determinante von R-Gittern modulo $(R^*)^2$ eindeutig bestimmt [Kne02, (1.13)].

(ii) Direkt aus der Definition des dualen Gitters folgt: L ist genau dann ganz, wenn $L \subseteq L^{\#}$.

(iii) Ein gerades Gitter L ist notwendigerweise ganz; denn seien $x, y \in L$, dann ist

$$b(x, y) = \frac{b(x + y, x + y) - b(x, x) - b(y, y)}{2} \in \mathbb{Z}.$$

(iv) Ist $B = (e_1, \ldots, e_n)$ eine Basis von L, dann ist $B^* := (e_1^*, \ldots, e_n^*)$ mit der Eigenschaft $b(e_i, e_j^*) = \delta_{ij}$ eine Basis von $L^{\#}$. Es gilt $\mathrm{Gram}(B^*) = \mathrm{Gram}(B)^{-1}$ [Kne02, (1.14)].

Da wir uns im Zuge dieser Arbeit in der Regel mit geraden Gittern quadratfreier Stufe beschäftigen, ist das folgende Lemma aus [Jü15, Lemma 1.1.1] von großer Bedeutung.

(2.1.6) Lemma

Sei L ein gerades Gitter der Stufe ℓ, wobei ℓ quadratfrei. Dann ist ℓ gleichzeitig die kleinste natürliche Zahl a, sodass $aL^{\#} \subseteq L$ gilt (also der Exponent der Diskriminantengruppe $L^{\#}/L$).

§ 2.2 Modulare Gitter

Wir kommen nun zum ursprünglich von Quebbemann eingeführten Konzept *modularer Gitter* [Que95]. Die hier verwendete Definition ist in [BFS05] zu finden.

(2.2.1) Definition

Sei L ein gerades Gitter und $\ell \in \mathbb{N}$.

(i) L heißt *ℓ-modular*, falls $L \cong \sqrt{\ell} L^{\#}$.

(ii) L heißt *stark ℓ-modular*, falls $L \cong \sqrt{m} L^{\#,m}$ für alle $m|l$, sodass $\mathrm{ggT}(m, \frac{\ell}{m}) = 1$.

(2.2.2) Lemma

Ist L ein gerades Gitter der Dimension n.

(i) Ist L ℓ-modular, dann ist $\mathrm{Det}(L) = \ell^{\frac{n}{2}}$. Insbesondere muss daher n gerade sein.

(ii) Ist L ℓ-modular und ℓ quadratfrei, dann hat L die Stufe ℓ.

(iii) Ist L stark ℓ-modular, von Stufe ℓ und ℓ quadratfrei, dann ist L auch ℓ-modular.

Beweis:

(i) Nach Bem. (2.1.5) ist $\mathrm{Det}(L^{\#}) = \mathrm{Det}(L)^{-1}$. Somit

$$\mathrm{Det}(L) = \mathrm{Det}(\sqrt{\ell}L^{\#}) = \ell^n \mathrm{Det}(L^{\#}) = \frac{\ell^n}{\mathrm{Det}(L)}.$$

Also folgt die Behauptung.

(ii) Sei a die Stufe von L, dann ist $\sqrt{a}L^{\#}$ gerade und hat insbesondere eine ganzzahlige Determinante. Nach (i) erhalten wir $\mathrm{Det}(\sqrt{a}L^{\#}) = \left(\frac{a^2}{\ell}\right)^{\frac{n}{2}} \overset{!}{\in} \mathbb{Z}$. Da ℓ quadratfrei ist, sieht man also $\ell | a$. Andersherum ist $L \cong \sqrt{\ell}L^{\#}$, also selbstverständlich auch $\sqrt{\ell}L^{\#}$ gerade, somit $a|\ell$.

(iii) L hat die quadratfreie Stufe ℓ, also ist $\ell L^{\#} \subseteq L$ nach Lemma (2.1.6). Wir erhalten

$$L \cong \sqrt{l}L^{\#,\ell} = \sqrt{\ell}\left(\frac{1}{\ell}L \cap L^{\#}\right) = \sqrt{\ell}L^{\#}. \qquad \square$$

Quebbemann zeigte in [Que95], dass die Theta-Reihen modularer Gitters Modulformen einer bestimmten Gruppe sind. Außerdem hat die Algebra der Modulformen eine besonders einfache Gestalt, wenn die Summe der Teiler von ℓ selbst ein Teiler von 24 ist. Konkret ist diese Eigenschaft für $\ell \in \{1, 2, 3, 5, 6, 7, 11, 14, 15, 23\}$ erfüllt. In der Literatur sind diese Stufen also besonders interessant. Es lässt sich zeigen (vgl. z.B. [Jü15, 1.2.2]), dass der Raum der Modulformen der erwähnten Gruppe in diesem Fällen ein eindeutiges Element θ der Form $1 + O(q^d)$ mit möglichst großem d und ganzzahligen Koeffizienten hat. Wir wollen den Begriff eines *extremalen Gitters* definieren als ein Gitter, welches ein möglichst großes Minimum besitzt, also ein Gitter mit Thetareihe θ. In unseren Spezialfällen gilt $d = 1 + \lfloor \frac{n}{k_1} \rfloor$, wobei k_1 Tabelle (2.1) zu entnehmen ist.

Wir können also definieren:

ℓ	1	2	3	5	6	7	11	14	15	23
k_1	24	16	12	8	8	6	4	4	4	2

Tabelle 2.1: k_1 Werte nach ℓ.

(2.2.3) Definition

Sei L ein ℓ-modulares Gitter der Dimension n und $\ell \in \{1, 2, 3, 5, 6, 7, 11, 14, 15, 23\}$. Erfüllt L die Schranke

$$\mathrm{Min}(L) \geq 2\left(1 + \lfloor \frac{n}{k_1} \rfloor\right),$$

wobei k_1 gewählt ist wie in Tabelle (2.1), so nennen wir L ein *extremales Gitter*.

Die Dimensionen, welche jeweils echt von k_1 geteilt werden bezeichnet man häufig auch als *Sprungdimensionen*, da in diesen Fällen das Minimum im Vergleich zur nächst kleineren Dimension um 2 nach oben "springt".

Da die Determinante für ℓ-modulare Gitter in fester Dimension nach Lemma (2.2.2) eindeutig bestimmt ist, liefern modulare Gitter mit möglichst großem Minimum die dichtesten Kugelpackungen. In diesem Sinne ist die Klassifikation extremaler Gitter besonders interessant.

(2.2.4) Definition

Die Funktion

$$\gamma : \{L \mid L \text{ ist } n\text{-dimensionales } \mathbb{Z}\text{-Gitter}\} \to \mathbb{R}, L \mapsto \frac{\mathrm{Min}(L)}{\mathrm{Det}(L)^{\frac{1}{n}}} \tag{2.1}$$

heißt *Hermite-Funktion*. Der Wert

$$\gamma_n := \max\{\gamma(L) \mid L \text{ ist } n\text{-dimensionales } \mathbb{Z}\text{-Gitter}\}$$

heißt *Hermite-Konstante* zur Dimension n.

Ein höherer Wert bezüglich der Funktion γ bedeutet dabei ein dichteres Gitter im Hinblick auf die dazugehörige Kugelpackung. In der Literatur wird häufig alternativ mit der sogenannten *Zentrumsdichte* $\delta(L) = \frac{\mathrm{Min}(L)^{\frac{n}{2}}}{2^n\sqrt{\mathrm{Det}(L)}}$ gearbeitet (vgl. [CS93, (1.5)]). Cohn und Elkies haben in [CE03] obere Schranken für die Zentrumsdichte ermittelt. Mithilfe der Identität $\gamma(L) = 4\delta(L)^{\frac{2}{n}}$ lassen sich daraus obere Schranken für die Hermite-Konstante herleiten. Zusätzlich sind für die Dimensionen 1 bis 8 und 24 die Werte von γ_n explizit bekannt. Hierfür können wir also die Hermite-Funktionen der dichtesten bekannten Gitter als Schranken festhalten (vgl. [NS]). Die sich ergebenden oberen Schranken in Dimensionen 1 bis 36 sind in Tabelle (2.2) festgehalten.

n	$\gamma_n \leq$	n	$\gamma_n \leq$	n	$\gamma_n \leq$	n	$\gamma_n \leq$
1	1	10	$2,2636$	19	3.3975	28	4.4887
2	1.1547	11	2.3934	20	3.5201	29	4.6087
3	1.2599	12	2.3934	21	3.6423	30	4.7286
4	1.4142	13	2.6494	22	3.7641	31	4.8484
5	1.5157	14	2.7759	23	3.8855	32	4.9681
6	1.6654	15	2.9015	24	4.0000	33	5.0877
7	1.8115	16	3.0264	25	4.1275	34	5.2072
8	2.0000	17	3.1507	26	4.2481	35	5.3267
9	2.1327	18	3.2744	27	4.3685	36	5.4462

Tabelle 2.2: Obere Schranken für γ_n bei $1 \leq n \leq 36$.

Diese Schranken sind sehr nützlich, da sie in vielen Fällen die Existenz von bestimmten Gittern von vorneherein ausschließt. Beispielsweise hätte ein hypothetisches extremales

23-modulares Gitter L in Dimension 6 bereits ein Minimum ≥ 8 und Determinante 23^3, also $\gamma(L) \geq \frac{8}{\sqrt{23}} \approx 1.6681 > 1.6654$ und kann somit nicht existieren. Genauer schließen die Schranken die folgenden extremalen Gitter aus:

(2.2.5) Lemma

Erfüllen $\ell \in \mathbb{N}$ und $1 \leq n \leq 36$ eine der Bedingungen

- $\ell = 1$ und $n \in \{2, 4, 6\}$,
- $\ell = 2$ und $n = 2$,
- $\ell = 11$ und $n \in \{20, 24, 28, 30, 32, 34, 36\}$,
- $\ell = 23$ und $n \in \{6, 8, 10, \ldots, 34, 36\}$,

so existiert kein extremales ℓ-modulares Gitter in Dimension n.

Beweis:

Tabelle (2.2). □

Vergleicht man die hypothetischen Zentrumsdichten extremaler Gitter - für die die Frage nach der Existenz bisher nach [Jü15] noch offen ist - mit denen der dichtesten bisher bekannten Gitter (zu finden in [NS]), so fällt auf, dass die Entdeckung extremaler Gitter in den folgenden Stufen ℓ und Dimensionen $1 \leq n \leq 48$ jeweils neue dichteste Kugelpackungen liefern würden:

- $\ell = 3$ und $n \in \{36, 38\}$.
- $\ell = 5$ und $n \in \{32, 36, 40, 44, 48\}$.
- $\ell = 6$ und $n = 40$.

- $\ell = 7$ und $n \in \{32, 34, 38, 40, 36\}$.

- $\ell = 11$ und $n \in \{18, 22\}$.

- $\ell = 14$ und $n = 28$.

- $\ell = 15$ und $n = 28$.

Aufgrund der dargelegten Beispiele wird deutlich, dass die Erforschung extremaler modularer Gitter von großem Interesse für die Gittertheorie ist. Im nächsten Kapitel beschreiben wir nun eine Vorgehensweise, modulare Gitter zu klassifizieren, welche zusätzlich eine Struktur als gebrochenes Ideal eines Zahlkörpers aufweisen - sogenannte *Ideal-Gitter*.

3 Ideal-Gitter

§ 3.1 Definitionen

Wir geben nun die Definition eines Ideal-Gitter abgeleitet aus [BFS05] an.

(3.1.1) Definition

(i) Ein *(algebraischer) Zahlkörper* ist eine endliche Erweiterung des Körpers $\mathbb{Q}$.

(ii) Der *Ring der ganzen Zahlen* eines Zahlkörpers K ist der Ring

$$\mathbb{Z}_K := \{a \in K \mid \mu_{a,\mathbb{Q}}(X) \in \mathbb{Z}[X]\}.$$

(iii) Die *Norm* eines Ideals $\mathcal{I}$ von $\mathbb{Z}_K$ ist definiert als

$$\mathcal{N}(\mathcal{I}) := |\mathbb{Z}_K/\mathcal{I}|.$$

(iv) Ein Zahlkörper K heißt *total-reell*, wenn für alle Einbettungen $\iota : K \to \mathbb{C}$ gilt, dass $\iota(K) \subseteq \mathbb{R}$ ist. Analog heißt ein Element α eines Zahlkörpers K total-reell, wenn für alle Einbettungen $\iota : K \to \mathbb{C}$ gilt, dass $\iota(\alpha) \in \mathbb{R}$ ist.

(v) Ein Zahlkörper K heißt *total-imaginär*, wenn für alle Einbettungen $\iota : K \to \mathbb{C}$ gilt, dass $\iota(K) \nsubseteq \mathbb{R}$ ist. Analog heißt ein Element α eines Zahlkörpers K total-imaginär, wenn für alle Einbettungen $\iota : K \to \mathbb{C}$ gilt, dass $\iota(\alpha) \notin \mathbb{R}$ ist.

(vi) Ein Zahlkörper K heißt *CM-Körper*, falls K total-imaginär ist und ein total-reeller Teilkörper $K^+ \leq K$ existiert mit $[K : K^+] = 2$.

(vii) Sei K ein CM-Körper und $\mathbb{Z}_K$ der Ring der ganzen Zahlen in K. Ein *Ideal-Gitter* ist ein Gitter $(\mathcal{I}, b)$, sodass $\mathcal{I}$ ein gebrochenes $\mathbb{Z}_K$-Ideal ist und $b : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ eine symmetrische positiv-definite Bilinearform mit $b(\lambda x, y) = b(x, \overline{\lambda} y)$ für $x, y \in \mathcal{I}$ und $\lambda \in \mathbb{Z}_K$. Die Abbildung $\lambda \mapsto \overline{\lambda}$ bezeichnet dabei die herkömmliche komplexe Konjugation.

(viii) Ein Element $\alpha \in K^+$ heißt *total-positiv*, für alle Einbettungen $\iota : K^+ \hookrightarrow \mathbb{R}$ gilt, dass $\iota(\alpha) > 0$ ist. Wir schreiben dann auch $\alpha \gg 0$. Die Menge aller total-positiven Elemente in K^+ wird mit $K^+_{\gg 0}$ bezeichnet.

Bis auf weiteres sei im Folgenden stets K ein CM-Körper, $\mathbb{Z}_K$ der Ring der ganzen Zahlen in K und K^+ der maximale total-reelle Teilkörper von K.

(3.1.2) Bemerkung

Die Eigenschaften der Bilinearform in der obrigen Definition sind nach [BFS05] äquivalent dazu, dass ein total-positives Element $\alpha \in K^+$ existiert, sodass die Bilinearform b die Gestalt $b(x, y) = \mathrm{Spur}_{K/\mathbb{Q}}(\alpha x \overline{y})$ annimmt. Wir können Ideal-Gitter daher auch durch die Notation $(\mathcal{I}, \alpha)$ beschreiben.

Ein Ideal-Gitter $\mathcal{I}$ kann immer auch als $\mathbb{Z}$-Gitter betrachtet werden, indem man $\mathbb{Z}_K$-Erzeuger von $\mathcal{I}$ und eine $\mathbb{Z}$-Basis von $\mathbb{Z}_K$ zu $\mathbb{Z}$-Erzeugern von $\mathcal{I}$ kombiniert. Im Folgenden bezeichnen wir daher $\mathcal{I}$ als gerade, ganz, modular, etc., falls $\mathcal{I}$ als $\mathbb{Z}$-Gitter diese Eigenschaften erfüllt und nennen $\mathcal{I}^\#$ das Dualgitter von $\mathcal{I}$ als $\mathbb{Z}$-Gitter.

Wir beschäftigen uns in dieser Arbeit mit Ideal-Gittern über zyklotomischen Zahlkörpern, also Körpern der Form $\mathbb{Q}(\zeta_m)$ für primitive m-te Einheitswurzeln ζ_m. Solche Körper sind CM-Körper mit maximalem total-reellem Teilkörper $K^+ = \mathbb{Q}(\zeta_m + \overline{\zeta_m})$. Wir erhalten Körper dieser Form, indem wir Automorphismen von $\mathbb{Z}$-Gittern betrachten, die wie primitive Einheitswurzeln operieren. Diese Aussagen wollen wir nun präzisieren. Dazu eine kurze Definition:

(3.1.3) Definition

Sei K ein Körper und $m \in \mathbb{N}$.

1. Ein Element $\zeta \in K$ heißt primitive m-te Einheitswurzel, falls $|\langle\zeta\rangle| = m$ ist.

2. Gilt $\mathrm{char}(K) \nmid m$ und sind $\zeta_1, \ldots, \zeta_n$ die primitiven m-ten Einheitswurzeln in einem Zerfällungskörper von $X^m - 1$, dann heißt das Polynom
$$\Phi_m(X) := \prod_{i=1}^{n} (X - \zeta_i)$$
das *m-te Kreisteilungspolynom.*

Einige wichtige bekannte Fakten zu Kreisteilungspolynomen (z.B. zu finden in [Mol11, Kap. 1]), sind die folgenden:

(3.1.4) Satz

(i) Gilt $\mathrm{char}(K) \nmid m$, so enthält der Zerfällungskörper von $X^m - 1$ genau $\varphi(m)$ primitive m-te Einheitswurzeln. Dabei ist $\varphi(m) := |\,(\mathbb{Z}/m\mathbb{Z})^*\,|$ die *Eulersche φ-Funktion.*

(ii) Ist $\mathrm{char}(K) = 0$, dann ist $\Phi_m \in \mathbb{Z}[X]$ und $X^m - 1 = \prod_{d|m} \Phi_d$.

(iii) Speziell für $K = \mathbb{Q}$ gilt $[\mathbb{Q}(\zeta_m) : \mathbb{Q}] = \varphi(m)$ und $\Phi_m \in \mathbb{Q}[X]$ ist irreduzibel.

(iv) Gilt $\mathrm{char}(K) \nmid m$, so ist $K(\zeta_m)/K$ ist eine Galoiserweiterung.

Wir sehen also, dass ζ_m genau dann eine primitive m-te Einheitswurzel ist, wenn sie das Minimalpolynom Φ_m hat. Wir können $\mathbb{Z}$-Gitter somit auf die folgende Weise als Ideal-Gitter auffassen (vgl. [Neb13, Abschnitt (5.2)]):

(3.1.5) Lemma
Sei L ein $\mathbb{Z}$-Gitter in einem n-dimensionalen bilinearen Vektorraum (V, b) und $\sigma \in \mathrm{Aut}(L)$ mit $\mu_\sigma = \Phi_m$ für ein $m \in \mathbb{N}$ mit $\varphi(m) = n$. Dann ist L isomorph zu einem Ideal-Gitter in $\mathbb{Q}(\zeta_m)$.

Beweis:
Durch die Operation von σ wird $\mathbb{Q}L$ mittels $\zeta_m \cdot x := \sigma(x)$ für $x \in \mathbb{Q}L$ zu einem eindimensionalen $\mathbb{Q}(\zeta_m)$-Vektorraum und L zu einem ein $\mathbb{Z}[\zeta_m]$-Modul. Wegen $\mathbb{Z}[\zeta_m] = \mathbb{Z}_{\mathbb{Q}[\zeta_m]}$ ist L also ein gebrochenes Ideal in $\mathbb{Q}(\zeta_m)$.
Da σ ein Automorphismus ist, ist die Bilinearform $b : L \times L \to \mathbb{Q}$ des Vektorraums ζ_m-invariant. Sei nun $\lambda \in \mathbb{Z}[\zeta_m]$ beliebig. Wir können $\lambda = \sum_{i=0}^{m-1} a_i \zeta_m^i$ für Koeffizienten $a_i \in \mathbb{Z}$ schreiben und sehen

$$b(\lambda x, y) = \sum_{i=0}^{m-1} a_i b(\zeta_m^i x, y) = \sum_{i=0}^{m-1} a_i b(x, \zeta_m^{-i} y) = \sum_{i=0}^{m-1} a_i b(x, \overline{\zeta_m^i} y) = b(x, \overline{\lambda} y),$$

womit die Eigenschaften eines Ideal-Gitters erfüllt sind. □

Mittels der Klassifikation der Ideal-Gitter über $\mathbb{Q}(\zeta_m)$ erhalten wir also zugleich alle $\mathbb{Z}$-Gitter mit Minimalpolynom Φ_m. Wie diese Klassifikation durchgeführt werden kann, erläutern wir in den nächsten Abschnitten.

§ 3.2 Strategie zur Klassifikation

Die in den nächsten Abschnitten beschriebenen Aussagen und Vorgehensweisen zur Klassifikation von Ideal-Gittern sind an [Jü15, Abschnitt (3.2)] und [Neb13, Abschnitt (5.2)] angelehnt.

(3.2.1) Definition
Das $\mathbb{Z}_K$-ideal

$$\Delta := \{x \in K \mid \mathrm{Spur}_{K/\mathbb{Q}}(x\overline{y}) \in \mathbb{Z} \text{ für alle } y \in \mathbb{Z}_K\}$$

bezeichnet die *inverse Differente* von $\mathbb{Z}_K$.

Wir können nun das Dual eines Idealgitters mithilfe der inversen Differente ausdrücken.

(3.2.2) Lemma
Sei $(\mathcal{I}, \alpha)$ ein Ideal-Gitter. Dann ist $\mathcal{I}^{\#} = \overline{\mathcal{I}}^{-1}\Delta\alpha^{-1}$ das Dualgitter von $\mathcal{I}$ als $\mathbb{Z}$-Gitter.

Beweis:

$$\begin{aligned}
\mathcal{I}^{\#} &= \{x \in K \mid b(x, \mathcal{I}) \subseteq \mathbb{Z}\} \\
&= \{x \in K \mid \mathrm{Spur}_{K/\mathbb{Q}}(\alpha x\overline{\mathcal{I}}) \subseteq \mathbb{Z}\} \\
&= \alpha^{-1}\{x \in K \mid \mathrm{Spur}_{K/\mathbb{Q}}(x\overline{\mathcal{I}}) \subseteq \mathbb{Z}\} \\
&= \alpha^{-1}\overline{\mathcal{I}}^{-1}\{x \in K \mid \mathrm{Spur}_{K/\mathbb{Q}}(x\overline{\mathbb{Z}_K}) \subseteq \mathbb{Z}\} \\
&= \overline{\mathcal{I}}^{-1}\Delta\alpha^{-1}.
\end{aligned}$$

□

Mit Blick auf modulare Gitter kann man damit die nächste Folgerung ziehen:

(3.2.3) Korollar

Sei ℓ quadratfrei und $(\mathcal{I}, \alpha)$ ein gerades Ideal-Gitter der Stufe ℓ. Die Menge $\mathcal{B} := \alpha\mathcal{I}\overline{\mathcal{I}}\Delta^{-1}$ ist ein $\mathbb{Z}_K$-Ideal mit $\ell\mathbb{Z}_K \subseteq \mathcal{B}$ und Norm $\mathcal{N}(\mathcal{B}) = \det(\mathcal{I})$.

Beweis:

Da ℓ quadratfrei ist, gilt $\ell\mathcal{I}^{\#} \subseteq \mathcal{I}$ nach Lemma (2.1.6). Mit Lemma (3.2.2) bedeutet dies:

$$\begin{aligned} & \ell\mathcal{I}^{\#} \subseteq \mathcal{I} \subseteq \mathcal{I}^{\#} \\ \Leftrightarrow \quad & \ell\overline{\mathcal{I}}^{-1}\Delta\alpha^{-1} \subseteq \mathcal{I} \subseteq \overline{\mathcal{I}}^{-1}\Delta\alpha^{-1} \\ \Leftrightarrow \quad & \ell\mathbb{Z}_K \subseteq \alpha\mathcal{I}\overline{\mathcal{I}}\Delta^{-1} \subseteq \mathbb{Z}_K. \end{aligned}$$

Für die Norm gilt

$$\det(\mathcal{I}) = |\mathcal{I}^{\#}/\mathcal{I}| = |\mathbb{Z}_K / \left(\mathcal{I}\left(\mathcal{I}^{\#}\right)^{-1}\right)| = |\mathbb{Z}_K/\mathcal{B}| = \mathcal{N}(\mathcal{B}). \qquad \square$$

Da es jeweils nur endlich viele $\mathbb{Z}_K$-Ideale mit bestimmter Norm gibt, existieren bei der Konstruktion von Idealgittern mit fester Determinante nur endlich viele Möglichkeiten für $\mathcal{B}$. Mithilfe der Primidealzerlegung lässt sich der rekursive Algorithmus (1) konstruieren, welcher alle Teiler eines Ideals $\mathcal{J}$ mit bestimmter Norm n berechnen kann.

Konkret wird unsere Strategie daraus bestehen, alle (relevanten) Möglichkeiten für $\mathcal{I}$ und $\mathcal{B}$ durchzugehen und zu testen, für welche davon das Ideal $\left(\mathcal{I}\overline{\mathcal{I}}\right)^{-1}\Delta\mathcal{B}$ ein Hauptideal mit total-positivem Erzeuger $\alpha \in K^{+}$ ist. In diesem Fall ist $(\mathcal{I}, \alpha)$ ein Ideal-Gitter. Um den Suchraum zu verkleinern, machen wir zunächst einige Einschränkungen.

Algorithmus 1 Berechnung aller Teiler mit fester Norm

Eingabe: $\mathbb{Z}_K$-Ideal $\mathcal{I}$, Norm n

Ausgabe: Liste aller Teiler von $\mathcal{I}$ mit Norm n

if $n = 1$ **then return** $[\mathbb{Z}_K]$

if $n \nmid \mathcal{N}(\mathcal{I}))$ **then return** $[\ ]$

if $\mathcal{N}(\mathcal{I}) = n$ **then return** $[\mathcal{I}]$

Zerlege $\mathcal{I}$ in Primideale $\mathcal{I} = \mathfrak{p}_1^{s_1} \dots \mathfrak{p}_k^{s_k}$

$n_\mathfrak{p} \leftarrow \mathcal{N}(\mathfrak{p}_1)$

$Results \leftarrow [\ ]$

for $j \in \{0, \dots s_1\}$ **do**

  if $n_\mathfrak{p}^j \mid n$ **then**

    $D \leftarrow$ Teiler von $\mathfrak{p}_2^{s_2} \dots \mathfrak{p}_k^{s_k}$ mit Norm $\frac{n}{n_\mathfrak{p}^j}$ (rekursiv)

    for $\mathcal{J} \in D$ **do**

      $Results \leftarrow Results \cup \left[\mathfrak{p}_1^j \mathcal{J}\right]$

return $Results$

§ 3.3 Klassengruppe

(3.3.1) Definition

Die *Klassengruppe*

$$\mathrm{Cl}_K := \{J \mid J \text{ ist gebrochenes } \mathbb{Z}_K\text{-Ideal}\}/\{(c)_{\mathbb{Z}_K} \mid c \in K^*\}.$$

(3.3.2) Lemma

Seien $\mathcal{I}$ ein gebrochenes $\mathbb{Z}_K$-Ideal und $\alpha \in K^+_{\gg 0}$. Für $\lambda \in K^*$ gilt $(\lambda\mathcal{I}, \alpha) \cong (\mathcal{I}, \lambda\overline{\lambda}\alpha)$.

Beweis:

Sei $b_\alpha : K \times K \to \mathbb{R}, (x, y) \mapsto \mathrm{Spur}_{K/\mathbb{Q}}(\alpha x\overline{y})$ die zu α gehörige Bilinearform. Dann ist

$$b_\alpha(\lambda x, \lambda y) = \mathrm{Spur}_{K/\mathbb{Q}}(\lambda\overline{\lambda}\alpha x\overline{y}) = b_{\lambda\overline{\lambda}\alpha}(x, y).$$

Folglich ist $\psi : (K, b_{\lambda\overline{\lambda}\alpha}) \to (K, b_\alpha), x \mapsto \lambda x$ eine Isometrie mit $\psi(\mathcal{I}) = (\lambda\mathcal{I})$. $\square$

Mit dieser Aussage genügt es also, aus jeder Klasse jeweils nur einen Vertreter zu betrachten. Wählt man $\lambda \in \mathbb{Z}_K^*$, so zeigt das Lemma, dass $(\mathcal{I}, \alpha) \cong (\mathcal{I}, \lambda\overline{\lambda}\alpha)$. Für α reichen also Vertreter modulo $\{\lambda\overline{\lambda} \mid \lambda \in \mathbb{Z}_K^*\}$.

Wir wollen nun die zu untersuchenden Möglichkeiten für $\mathcal{I}$ noch weiter einschränken: Ist $K/\mathbb{Q}$ galoissch (wie es für zyklotomische Zahlkörper der Fall ist), so genügt ein Repräsentant modulo der Operation der Galosgruppe.

(3.3.3) Lemma

Sei $K/\mathbb{Q}$ eine Galoiserweiterung, $\mathcal{I}$ ein gebrochenes $\mathbb{Z}_K$-Ideal und $\alpha \in K^+_{\gg 0}$. Für $\sigma \in \mathrm{Gal}(K/\mathbb{Q})$ ist $(\mathcal{I}, \alpha) \cong (\sigma(\mathcal{I}), \sigma(\alpha))$.

Beweis:

Da die Spur invariant unter der Galoisgruppe ist, erhält man die folgende Gleichungskette.

$$\begin{aligned} b_{\sigma(\alpha)}(\sigma(x), \sigma(y)) &= \mathrm{Spur}_{K/\mathbb{Q}} \left(\sigma(\alpha)\sigma(x)\overline{\sigma(y)}\right) \\ &= \mathrm{Spur}_{K/\mathbb{Q}} \left(\sigma\left(\alpha x \overline{y}\right)\right) = \mathrm{Spur}_{K/\mathbb{Q}} \left(\alpha x \overline{y}\right) = b_\alpha(x, y) \end{aligned}$$

Also induziert σ eine Isometrie $\sigma : (K, b_\alpha) \to (K, b_{\sigma(\alpha)})$. □

(3.3.4) Bemerkung

Mit `MAGMA` kann die Klassengruppe berechnet werden, in gewissen Fällen ist der zugehörige Algorithmus jedoch sehr ressourcenintensiv. Unter Annahme der unbewiesenen *verallgemeinerten Riemann-Hypothese* erlauben gewisse Schranken eine leichtere Berechnung. Für die Implementierung gehen wir daher von der Korrektheit dieser Hypothese aus. Sollte diese falsch sein, so sind die später angeführten Listen der gefundenen modularen Gitter möglicherweise unvollständig.

§ 3.4 Total-positive Erzeuger

Wir benötigen nun einen Test, welcher für ein gegebenes gebrochenes $\mathbb{Z}_K$-Ideal $\mathcal{I}$ überprüft, dieses von einem total-positiven Element $\alpha \in K^+_{\gg 0}$ erzeugt wird. Dazu untersuchen wir zuerst, ob $\mathcal{I}$ überhaupt von einem Element aus K^+ erzeugt ist und anschließend, wann ein Ideal $\alpha'\mathbb{Z}_K$ für $\alpha' \in K^+$ einen total-positiven Erzeuger hat.

(3.4.1) Satz

Sei $\mathcal{I}$ ein gebrochenes $\mathbb{Z}_K$-Ideal. Es existiert genau dann ein $\alpha' \in K^+$ mit $\mathcal{I} = \alpha'\mathbb{Z}_K$, wenn $\mathcal{I} \cap K^+ = \alpha'\mathbb{Z}_{K^+}$ und für jeden Primteiler $\mathfrak{p}$ von $\mathcal{I}$ gilt

- Ist $\mathfrak{p}$ verzweigt in K/K^+, so ist $\nu_{\mathfrak{p}}(\mathcal{I}) \in 2\mathbb{Z}$.
- Ist $\mathfrak{p}$ unverzweigt in K/K^+, so ist $\nu_{\mathfrak{p}}(\mathcal{I}) = \nu_{\overline{\mathfrak{p}}}(\mathcal{I})$.

Beweis:

Wir zeigen zunächst, dass die Bedingungen an die Primteiler äquivalent dazu sind, dass $\mathcal{I} = (\mathcal{I} \cap K^+)\mathbb{Z}_K$.

Sei dazu zuerst $\mathcal{I} = (\mathcal{I} \cap K^+)\mathbb{Z}_K$ erfüllt. Seien

$$\mathcal{I}' := \mathcal{I} \cap K^+ = \prod_{\mathfrak{a}} \mathfrak{a}^{\nu_{\mathfrak{a}}(\mathcal{I} \cap K^+)}, \quad \mathcal{I} = \prod_{\mathfrak{p}} \mathfrak{p}^{\nu_{\mathfrak{p}}(\mathcal{I})}$$

die Primidealzerlegungen. Dann folgt

$$\prod_{\mathfrak{a}} \mathfrak{a}^{\nu_{\mathfrak{a}}(\mathcal{I}')}\mathbb{Z}_K = \prod_{\mathfrak{p}} \mathfrak{p}^{\nu_{\mathfrak{p}}(\mathcal{I})}.$$

Aufgrund der Eindeutigkeit der Primidealzerlegung bedeutet dies

$$\mathfrak{a}^{\nu_{\mathfrak{a}}(\mathcal{I}')}\mathbb{Z}_K = \prod_{\mathfrak{p}|\mathfrak{a}} \mathfrak{p}^{\nu_{\mathfrak{p}}(\mathcal{I})}$$

für jedes Primideal $\mathfrak{a}$ von $\mathbb{Z}_{K^+}$. Es ist $[K : K^+] = 2$, also kann $\mathfrak{a}\mathbb{Z}_K$ nur eine der Formen $\mathfrak{p}$, $\mathfrak{p}^2$, oder $\mathfrak{p}\overline{\mathfrak{p}}$ für ein Primideal $\mathfrak{p}$ in $\mathbb{Z}_K$ annehmen.

- Falls $\mathfrak{a}\mathbb{Z}_K = \mathfrak{p}^2$ (also falls $\mathfrak{p}$ verzweigt ist), so folgt $\nu_{\mathfrak{p}}(\mathcal{I}) = 2\nu_a(\mathcal{I}') \in 2\mathbb{Z}$.
- In den anderen beiden Fällen (also falls $\mathfrak{p}$ unverzweigt ist) gilt $\nu_{\mathfrak{p}}(\mathcal{I}) = \nu_{\mathfrak{a}}(\mathcal{I}') = \nu_{\mathfrak{p}}(\overline{\mathcal{I}})$.

Seien nun andersherum die Primideal-Bedingungen erfüllt. Definiert man

$$\mathcal{I}' := \prod_{\mathfrak{a}} \mathfrak{a}^{\nu_{\mathfrak{a}}(\mathcal{I}')}, \quad \nu_{\mathfrak{a}}(\mathcal{I}') = \begin{cases} \nu_{\mathfrak{p}}(\mathcal{I}) & \mathfrak{a}\mathbb{Z}_K \in \{\mathfrak{p}, \mathfrak{p}\overline{\mathfrak{p}}\} \\ \frac{1}{2}\nu_{\mathfrak{p}}(\mathcal{I}) & \mathfrak{a}\mathbb{Z}_K = \mathfrak{p}^2 \end{cases}$$

so gilt

$$\mathcal{I} = \prod_{\mathfrak{p}} \mathfrak{p}^{\nu_{\mathfrak{p}}(\mathcal{I})} = \prod_{\mathfrak{a}} (\mathfrak{a}\mathbb{Z}_K)^{\nu_{\mathfrak{a}}(\mathcal{I}')} = \mathcal{I}'\mathbb{Z}_K.$$

Also folgt $(\mathcal{I} \cap K^+)\mathbb{Z}_K = (\mathcal{I}'\mathbb{Z}_K \cap K^+)\mathbb{Z}_K = \mathcal{I}'\mathbb{Z}_K = \mathcal{I}$ und es gilt die behauptete Äquivalenz.

Die Behauptung des Satzes wurde somit darauf reduziert, dass genau dann $\mathcal{I} = \alpha'\mathbb{Z}_K$, wenn $\mathcal{I} \cap K^+ = \alpha'\mathbb{Z}_{K^+}$ und $\mathcal{I} = (\mathcal{I} \cap K^+)\mathbb{Z}_K$ für $\alpha' \in K^+$. Dies folgt allerdings leicht mithilfe von $(\alpha'\mathbb{Z}_K) \cap K^+ = \alpha'\mathbb{Z}_{K^+}$. □

Mithilfe dieses Satzes können wir nun Algorithmus (2) formulieren, welcher zu einem gegebenen Ideal $\mathcal{I}$ testet, ob dieses einen Erzeuger in K^+ hat und - falls ja - einen solchen zurückgibt. Ein Primideal $\mathfrak{a}$ wie im Beweis unseres Satzes erhalten wir, indem wir eine Primzahl p finden, sodass $\mathfrak{p} | (p\mathbb{Z}_K)$, dann muss $\mathfrak{a}$ eines der Primideale aus der Faktorisierung von $p\mathbb{Z}_{K^+}$ teilen.

(3.4.2) Lemma
Sei $\alpha' \in K^+$. Ein total-positives Element $\alpha \in K^+_{\gg 0}$ ist genau dann ein Erzeuger des Ideals $\alpha'\mathbb{Z}_K$, wenn eine Einheit $\epsilon \in \mathbb{Z}^*_{K^+}$ existiert mit $\alpha = \alpha'\epsilon$ und $\operatorname{sign}((\iota(\epsilon)) = \operatorname{sign}(\iota(\alpha'))$ für alle Einbettungen $\iota : K^+ \hookrightarrow \mathbb{R}$.

Beweis:
Ein weiterer Erzeuger hat immer die Gestalt $\alpha = \alpha'\epsilon$ für eine Einheit $\epsilon \in (\mathbb{Z}_{K^+})^*$. Damit α total-positiv wird muss für alle Einbettungen $\iota : K^+ \hookrightarrow \mathbb{R}$ gelten:

$$1 \stackrel{!}{=} \operatorname{sign}(\iota(\alpha)) = \operatorname{sign}\Big(\iota(\alpha')\iota(\epsilon)\Big) = \operatorname{sign}\Big(\iota(\alpha')\Big) \operatorname{sign}(\iota(\epsilon))$$

Algorithmus 2 Berechnung eines total-reellen Erzeugers

Eingabe: $\mathbb{Z}_K$-Ideal $\mathcal{I}$

Ausgabe: Element $\alpha' \in K^+$ mit $\alpha'\mathbb{Z}_K = \mathcal{I}$, oder *false*, falls ein solches Element nicht existiert

$\mathcal{I}' \leftarrow 1\mathbb{Z}_{K^+}$
Split $\leftarrow [\ ]$
Zerlege $\mathcal{I} = \mathfrak{p}_1^{s_1} \dots \mathfrak{p}_k^{s_k}$ in Primideale.
for $i \in \{1 \dots k\}$ **do**
    if $i \in$ Split **then continue**
    $p \leftarrow$ Minimale natürliche Zahl $p \in \mathbb{N}$ mit $\mathfrak{p}_i | (p\mathbb{Z}_K)$
    Zerlege $p\mathbb{Z}_{K^+}$ in Primideale: $p\mathbb{Z}_{K^+} = \mathfrak{q}_1^{t_1} \dots \mathfrak{q}_l^{t_l}$
    $\mathfrak{a} \leftarrow \mathfrak{q}_j$ mit $\mathfrak{p}_i | (\mathfrak{q}_j\mathbb{Z}_K)$
    if $\mathfrak{a}\mathbb{Z}_K = \mathfrak{p}_i^2$ **then**
        if $2 \nmid s_i$ **then return** *false*
        $\mathcal{I}' \leftarrow \mathcal{I}'\mathfrak{a}^{\frac{s_i}{2}}$
    else if $\mathfrak{a}\mathbb{Z}_K = \mathfrak{p}_i$ **then**
        $\mathcal{I}' \leftarrow \mathcal{I}'\mathfrak{a}^{s_i}$
    else if $\mathfrak{a}\mathbb{Z}_K = \mathfrak{p}_i\overline{\mathfrak{p}_i}$ **then**
        if $\nu_{\mathfrak{p}_i}(\mathcal{I}) \neq \nu_{\overline{\mathfrak{p}_i}}(\mathcal{I})$ **then return** *false*
        $\mathcal{I}' \leftarrow \mathcal{I}'\mathfrak{a}^{s_i}$
        $j \leftarrow j'$ mit $\mathfrak{p}_{j'} = \overline{\mathfrak{p}_i}$
        Split $\leftarrow$ Split $\cup [j]$
if $\mathcal{I}'$ kein Hauptideal **then**
    return *false*
else
    return Erzeuger von $\mathcal{I}'$

Also müssen die Vorzeichen jeweils identisch sein. $\square$

Elemente aus $\left(\mathbb{Z}_{K^+}^*\right)^2$ haben immerzu ein positives Signum bezüglich aller Einbettungen. Außerdem liefern total-positive Elemente, die in der gleichen Klasse modulo Quadraten liegen, nach Lemma (3.3.2) isomorphe Idealgitter. Es genügt also, sich bei der Suche nach einer Einheit wie im vorherigen Lemma auf Vertreter modulo Quadraten zu beschränken. Nach dem Dirichletschen Einheitensatz [Neu92, Theorem (7.4)] hat die Einheitengruppe die Struktur

$$\mathbb{Z}_{K^+}^* = \{\pm 1\} \times \mathbb{Z}^{t-1}$$

mit $t := [K^+ : \mathbb{Q}]$. Die Erzeuger $(\epsilon_1, \ldots \epsilon_t)$ der Gruppe heißen *Grundeinheiten*. Jede Einheit ϵ lässt sich also darstellen in der Form $\epsilon = \epsilon_1^{\nu_1} \ldots \epsilon_t^{\nu_t}$. Das folgende Korollar liefert uns nun die Lösung auf unsere Frage nach den total-positiven Erzeugern.

Dann lässt sich folgendes Korollar ziehen:

(3.4.3) Korollar

Sei $\alpha' \in K^+$, seien die Einbettungen von K^+ in $\mathbb{R}$ gegeben durch $\iota_1, \ldots, \iota_t$ und seien $\epsilon_1, \ldots, \epsilon_t$ die Grundeinheiten von $\mathbb{Z}_{K^+}^*$. Definiere die Matrix

$$M \in \mathbb{F}_2^{t \times t}, \quad M_{ij} = \begin{cases} 1 & , \mathrm{sign}(\iota_j(\epsilon_i)) = -1 \\ 0 & , \mathrm{sign}(\iota_j(\epsilon_i)) = 1 \end{cases}$$

und den Vektor

$$V \in \mathbb{F}_2^{1 \times t}, \quad V_i = \begin{cases} 1 & , \mathrm{sign}(\iota_i(\alpha')) = -1 \\ 0 & , \mathrm{sign}(\iota_i(\alpha')) = 1. \end{cases}$$

Dann sind die total-positiven Erzeuger des Ideals $\alpha'\mathbb{Z}_K$ genau die Elemente der Menge $\{\alpha' \epsilon_1^{x_1} \ldots \epsilon_t^{x_t} \epsilon^2 \mid x \in \mathbb{F}_2^{1 \times t}, xM = V, \epsilon \in (\mathbb{Z}_{K^+}^*)\}$.

Beweis:

Nach Lemma (3.4.2) und da Quadrate immerzu ein positives Signum haben, sind die total-positiven Erzeuger gegeben durch die Elemente $u = \alpha' \epsilon_1^{x_1} \dots \epsilon_t^{x_t} \epsilon^2$, wobei $x \in \mathbb{F}_2^{1 \times t}$, $\epsilon \in \mathbb{Z}_{K^+}^*$ und $\epsilon_1^{x_1} \dots \epsilon_t^{x_t}$ bezüglich aller Einbettungen dasselbe Signum wie α' hat. Das Signum bezüglich einem ι_i ist genau dann gleich, wenn

$$|\{j \mid \mathrm{sign}(\iota_j(\epsilon_i)) = -1 \text{ und } x_i = 1\}| \equiv \begin{cases} 1 \pmod 2 & , \mathrm{sign}(\iota_j(\alpha')) = -1 \\ 0 \pmod 2 & , \mathrm{sign}(\iota_j(\alpha')) = 1 \end{cases}.$$

Diese Kongruenz ist aber genau dann erfüllt, wenn x Lösung des linearen Gleichungssystems $xM = V$ ist. □

(3.4.4) Bemerkung

Um später in der Implementierung Zeit zu sparen, kann man bemerken, dass sich verschiedene total-positive Erzeuger des gleichen Ideals jeweils lediglich um eine total-positive Einheit unterscheiden. Um die Effizienz zu erhöhen, kann man also zu Beginn des Algorithmus die Menge aller total-positiven Einheiten (diese korrespondieren zum Kern von M) berechnen. Auf diese Weise muss später pro Ideal jeweils nur eine spezielle Lösung des Gleichungssystems gefunden werden und die Menge aller total-positiven Erzeuger lässt sich durch Multiplikation mit den vorher berechneten total-positiven Einheiten erstellen.

Eine weitere Anmerkung zur Implementierung: `MAGMA` kann mit der Funktion `pFundamentalUnits` eine Untergruppe von $\mathbb{Z}_{K^+}^*$ mit ungeradem Index berechnen. Wie das folgende Lemma zeigt, reicht dies für unser Vorhaben bereits aus, da wir nur ein Vertretersystem der Einheiten modulo Quadraten benötigen.

(3.4.5) Lemma

Sei G eine abelsche Gruppe und $U \leq G$ mit $[G : U]$ ungerade. Dann ist

$$G/G^2 \cong U/U^2.$$

Beweis:
Betrachte den Epimorphismus $\pi : G \to G/G^2$. Es ist bereits $\pi|_U$ surjektiv, denn sei $gG^2 \in G/G^2$, dann ist $g^{[G:U]} \in U$, da $(gU)^{[G:U]} = U$ und - weil der Index ungerade ist - auch $\pi\left(g^{[G:U]}\right) = gG^2$. Zudem ist $\text{Kern}(\pi|_U) = U \cap G^2 = U^2$, denn für $g^2 \in U \cap G^2$ muss $|gU| \leq 2$ gelten. Die Ordnung kann aber wegen des ungeraden Index nicht 2 sein, also folgt bereits $g \in U$ und somit $g^2 \in U^2$. Mit dem Homomorphiesatz ergibt sich nun die Behauptung. $\square$

Anhand der gewonnenen Erkenntnisse erstellen wir nun einen Algorithmus (3), der zu einem Ideal $\mathcal{I} = \alpha'\mathbb{Z}_K$ für $\alpha' \in K^+$ ein Vertretersystem aller total-positiven Erzeuger $\alpha \in K^+_{\gg 0}$ modulo $\lambda\overline{\lambda}$ für $\lambda \in \mathbb{Z}_K^*$ liefert. Die Ergebnisse der Zeilen $6-14$ können in der Implementierung nach einmaliger Durchführung abgespeichert werden, sodass die Resultate anschließend für jedes zu prüfende α' wiederverwertet werden können.

Algorithmus 3 Berechnung total-positiver Erzeuger

Eingabe: Erzeuger $\alpha' \in K^+$ von $\mathcal{I}$
Ausgabe: Liste von Vertretern der Menge aller total-positiven Erzeuger $\alpha \in K^+_{\gg 0}$ von $\mathcal{I}$ modulo $\{\lambda\overline{\lambda} \mid \lambda \in \mathbb{Z}_K^*\}$ zurückgibt

$\iota_1, \ldots, \iota_t \leftarrow$ Einbettungen $K^+ \hookrightarrow \mathbb{R}$
$\epsilon_1, \ldots \epsilon_t \leftarrow$ Erzeuger einer Untergruppe von $\mathbb{Z}_{K^+}^*$ mit ungeradem Index
$M \leftarrow 0 \in \mathbb{F}_2^{t \times t}$
for $(i, j) \in \underline{t} \times \underline{t}$ **do**
    if $\iota_j(\epsilon_i) < 0$ **then**
        $M_{ij} \leftarrow 1$
$U' \leftarrow [\epsilon_1^{a_1} \ldots \epsilon_t^{a_t} \mid a \in \mathrm{Kern}(M)]$
$U \leftarrow [\,]$
for $u' \in U'$ **do**
    if $u' \neq u\lambda\overline{\lambda}$ für alle $u \in U, \lambda \in \mathbb{Z}_K^*$ **then**
        $U \leftarrow U \cup [u']$
$V \leftarrow 0 \in \mathbb{F}_2^{1 \times t}$
for $i \in \{1, \ldots, t\}$ **do**
    if $\iota_i(\alpha') < 0$ **then**
        $V_i \leftarrow 1$
$x \leftarrow$ Lösung von $xM = V$
return $\alpha'\epsilon_1^{x_1} \ldots \epsilon_t^{x_t} U$

§ 3.5 Finaler Algorithmus und Ergebnisse

Alle bisherigen Bestandteile können nun zu einem Algorithmus zusammengesetzt werden, der zu einem quadratfreien $\ell \in \mathbb{N}$, einer vorgegebenen Determinante d und einem

CM-Körper K mit total-reellem Teilkörper K^+ alle Ideal-Gitter berechnet.

Algorithmus 4 Berechnung von Ideal-Gittern

Eingabe: Quadratfreies $\ell \in \mathbb{N}$, $d \in \mathbb{N}$, CM-Körper K, maximaler total-reeller Teilkörper K^+ von K

Ausgabe: Per Isomorphie reduzierte Liste aller geraden Ideal-Gitter $(\mathcal{I}, \alpha)$ über K mit Determinante d, deren Stufe ℓ teilt

$\mathfrak{A} \leftarrow$ Vertretersystem von $Cl_K / \mathrm{Gal}(K/\mathbb{Q})$

$\mathfrak{B} \leftarrow [\mathcal{B} \mid \mathcal{B}$ ist $\mathbb{Z}_K$-Ideal mit $\ell\mathbb{Z}_K \subseteq \mathcal{B} \subseteq \mathbb{Z}_K$ und $\mathcal{N}(\mathcal{B}) = d]$ (nach Algorithmus (1))

$Results \leftarrow [\ ]$

for $(\mathcal{I}, \mathcal{B}) \in (\mathfrak{A}, \mathfrak{B})$ **do**

 $\mathcal{J} \leftarrow \left(\mathcal{I}\overline{\mathcal{I}}\right)^{-1} \Delta\mathcal{B}$

 if $\exists \alpha' \in K^+$ mit $\mathcal{J} = \alpha'\mathbb{Z}_K$ (nach Algorithmus (2)) **then**

  $X \leftarrow \left[\alpha \in K^+_{\gg 0} \mid \mathcal{J} = \alpha\mathbb{Z}_K\right]$ (nach Algorithmus (3))

  for $\alpha \in X$ **do**

   if $(\mathcal{I}, \alpha)$ ist gerades Gitter **then**

    if $(\mathcal{I}, \alpha) \ncong (\tilde{\mathcal{I}}, \tilde{\alpha})$ für alle $(\tilde{\mathcal{I}}, \tilde{\alpha}) \in Results$ **then**

     $Results \leftarrow Results \cup [(\mathcal{I}, \alpha)]$

Mit diesem Algorithmus kann man nun alle ℓ-modularen Gitter in Dimension n klassifizieren, welche einen Automorphismus σ besitzen mit $\mu_\sigma = \Phi_m$ und $\varphi(m) = n$. Dazu wendet man Algorithmus (4) wie in Lemma (2.2.2) und Lemma (3.1.5) besprochen mit $d = l^{\frac{n}{2}}$ und $K = \mathbb{Q}(\zeta_m)$ an. Eine weitere kleine Erleichterung bringt in diesem Spezialfall die Tatsache, dass $\mathbb{Q}(\zeta_m) \cong \mathbb{Q}(\zeta_{2m})$, falls $m \equiv 1 \pmod 2$. Insbesondere sind die Ideal-Gitter über $\mathbb{Q}(\zeta_m)$ und $\mathbb{Q}(\zeta_{2m})$ dieselben. Man kann also für eine vollständige Aufzählung alle m mit $m \equiv 2 \pmod 4$ weglassen. Eine Implementierung in `MAGMA` liefert nun alle ℓ-modularen Ideal-Gitter mit Dimension $n \leq 36$ (eine Steigerung der Dimension

n \ ℓ	1	2	3	5	6	7	11	14	15	23
4	–	1(1)	1(1)	–	–	–	1(1)	1(1)	–	1(1)
6	–	–	1(1)	–	–	1(1)	–	–	–	–
8	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	2(1)	2(1)	1(1)	3(–)
10	–	–	–	–	–	–	1(1)	–	–	–
12	–	1(1)	2(1)	1(1)	1(1)	1(–)	1(–)	1(1)	–	1(–)
16	1(1)	2(1)	3(2)	1(–)	2(1)	4(3)	5(–)	5(–)	3(1)	5(–)
18	–	–	1(–)	–	–	–	–	–	–	–
20	–	1(1)	–	–	1(1)	1(–)	2(–)	–	–	–
22	–	–	–	–	–	–	–	–	–	2(–)
24	4(1)	2(1)	7(1)	5(1)	5(2)	8(–)	7(–)	8(–)	5(–)	14(–)
32	7(5)	13(4)	13(7)	10(–)	12(–)	19(–)	42(–)	21(–)	23(–)	–
36	–	6(3)	8(–)	8(–)	–	–	2(–)	36(–)	4(–)	–

Tabelle 3.1: Anzahl der ℓ-modularen Ideal-Gitter in Dimension $n \leq 36$ sowie der Anzahl der extremalen Gitter darunter

beansprucht exponentiell höheren Zeitaufwand) und $\ell \in \{1, 2, 3, 5, 6, 7, 11, 14, 15, 23\}$. In Tabelle (3.5) sind die Gesamtzahlen der Ideal-Gitter zu finden; außerdem befindet sich in Anhang (6.1) eine ausführlichere Zusammenfassung der Klassifikationsergebnisse mit zusätzlicher Angabe der zugrundeliegenden zyklotomischen Zahlkörpern und der Anzahl der Gitter, aufgeschlüsselt nach Minimum. Beachte dabei: Der Zahlkörper ist im Allgemeinen nicht eindeutig, ein Gitter kann möglicherweise Ideal-Gitter-Struktur über mehreren Kreisteilungskörpern gleichzeitig aufweisen; in der Tabelle im Anhang ist jeweils nur einer davon genannt.

4 Sub-Ideal-Gitter

§ 4.1 Einführung

Im letzten Kapitel haben wir gesehen, wie Gitter in Dimension n mit einem Automorphismus σ klassifiziert werden können, falls $\mu_\sigma = \Phi_m$ und $\varphi(m) = n$ erfüllt sind. In diesem Kapitel wollen wir versuchen, Aussagen über Gitter L zu treffen, welche nicht selbst Ideal-Gitter-Struktur aufweisen, aber zumindest ein Ideal-Gitter enthalten.

Ist L ein Gitter und $\sigma \in \mathrm{Aut}(L)$ von endlicher Ordnung mit Minimalpolynom $\mu_\sigma = \Phi_{m_1} \dots \Phi_{m_k}$, so können wir den zugrundeliegenden Vektorraum V in σ-invariante Teilräume aufspalten:

$$V = \mathrm{Kern}(\Phi_{m_1}(\sigma)) \oplus \cdots \oplus \mathrm{Kern}(\Phi_{m_k}(\sigma)).$$

Wie das folgende Lemma zeigt, ist diese Zerlegung sogar orthogonal:

(4.1.1) Lemma
Sei (V, b) ein bilinearer Vektorraum und $\sigma \in O(V, b)$ mit $\mu_\sigma = \Phi_{m_1}\Phi_{m_2} \dots \Phi_{m_k}$, dann ist

$$V = \mathrm{Kern}(\Phi_{m_1}(\sigma)) \perp \mathrm{Kern}(\Phi_{m_2}(\sigma)) \perp \cdots \perp \mathrm{Kern}(\Phi_{m_k}(\sigma))$$

eine Zerlegung in σ-invariante, orthogonale Teilräume.

Beweis:
Seien $\pi_i : V \twoheadrightarrow \operatorname{Kern}(\Phi_{m_i}(\sigma))$ für $i = 1, \ldots, k$ die Projektionen auf die Komponenten. Da σ eine Isometrie ist, gilt $\sigma^{ad} = \sigma^{-1}$, also ist σ insbesondere normal und damit auch die π_i, welche sich als Polynome in σ mit rationalen Koeffizienten ausdrücken lassen. Normale Projektionen sind allerdings selbstadjungiert, also gilt für alle $x, y \in V$ sowie $i \neq j$:

$$b(\pi_i(x), \pi_j(y)) = b(x, \pi_i^{ad}(\pi_j(y))) = b(x, \pi_i(\pi_j(y))) = b(x, 0) = 0$$

und daher $\pi_i(V) \perp \pi_j(V)$. Die angegebene Zerlegung von V ist somit orthogonal. $\square$

Besitzt μ_σ einen Teiler Φ_m, wobei $\frac{n}{2} < \varphi(m) \leq n$, so muss $\operatorname{Kern}(\Phi_m(\sigma))$ die Dimension $\varphi(m)$ haben, wird also zu einem eindimensionalen $\mathbb{Q}(\zeta_m)$-Vektorraum. Die orthogonale Zerlegung des Vektorraums induziert also ein volles Teilgitter

$$M := (L \cap \operatorname{Kern}(\Phi_m(\sigma))) \perp \left(L \cap \operatorname{Kern}\left(\frac{\mu_\sigma}{\Phi_m}(\sigma)\right)\right) \leq L$$

und $L \cap \operatorname{Kern}(\Phi_m(\sigma))$ hat eine Struktur als Ideal-Gitter über dem zyklotomischen Zahlkörper $\mathbb{Q}(\zeta_m)$. Vergleiche dazu [Neb13, Abs. (5.3)]. Dies halten wir in der folgenden Definition fest.

(4.1.2) Definition
Sei L ein $\mathbb{Z}$-Gitter der Dimension n.

(i) Ein *großer Automorphismus* von L ist ein $\sigma \in \operatorname{Aut}(L)$ von Ordnung $m \in \mathbb{N}$, sodass $\Phi_m | \mu_\sigma$ und $\frac{n}{2} < \varphi(m) \leq n$.

(ii) Ist $\sigma \in \operatorname{Aut}(L)$ ein großer Automorphismus, so bezeichnet man das Ideal-Gitter $L \cap \operatorname{Kern}(\Phi_m(\sigma))$ über $\mathbb{Q}(\zeta_m)$ als *Sub-Ideal-Gitter* von L.

Unsere Strategie zur Klassifikation der Gitter L besteht darin, Eigenschaften des induzierten Teilgitters M zu zeigen sowie die Operation von σ genauer zu untersuchen, um eine Liste möglicher Kandidaten für M und σ zu finden. Anschließend konstruieren wir L als σ-invariantes Obergitter von M.

Für Gitter mit großen Automorphismen können wir die Ideal-Gitter-Komponente mithilfe der Algorithmen aus dem letzten Kapitels effizient konstruieren. Probleme bereitet uns allerdings der andere Teil Kern $\left(\frac{\mu_\sigma}{\Phi_m}(\sigma)\right)$ des Vektorraums, über welchen wir a priori nicht viel aussagen können. Abhilfe schaffen uns unter gewissen Umständen die Automorphismen von Primzahlordnung.

§ 4.2 Automorphismen von Primzahlordnung

Der folgende Abschnitt ist an [Jü15, Kap. 4] und [Neb13, Kap. 4] angelehnt.

Sei L in diesem Abschnitt ein $\mathbb{Z}$-Gitter in einem n-dimensionalen bilinearen $\mathbb{Q}$-Vektorraum (V, b) und $\sigma \in \mathrm{Aut}(L)$ von Primzahlordnung p. Dann ist $\mu_\sigma \in \{\Phi_p, \Phi_1\Phi_p\}$. Wie vorher erhält man eine σ-invariante Zerlegung

$$V = \mathrm{Kern}(\Phi_1(\sigma)) \perp \mathrm{Kern}(\Phi_p(\sigma)) =: V_1 \oplus V_p.$$

Es ist $\Phi_1(X) = X - 1$, also $V_p = \mathrm{Bild}(\sigma - 1)$ und $V_1 = \mathrm{Kern}(\sigma - 1)$. Seien n_p die Dimension von V_p und n_1 die Dimension von V_1 über $\mathbb{Q}$. Da V_p eine Struktur als $\mathbb{Q}(\zeta_p)$-Vektorraum hat und $\mathrm{Dim}_{\mathbb{Q}}(\mathbb{Q}(\zeta_p)) = p - 1$, muss n_p von $p - 1$ geteilt werden.

Die durch die orthogonale Zerlegung von V induzierten Gitter $L_1 := L \cap V_1$ und $L_p := L \cap V_p$ nennen wir das von σ induzierte *Fix-Gitter* und das von σ induzierte *Bild-Gitter*. Außerdem sei $M := L_1 \perp L_p \leq L$. Im Folgenden wollen wir die Struktur von M näher untersuchen.

(4.2.1) Lemma

Seien σ, L_1 und L_p wie oben definiert.

(i) Es existiert ein Polynom $v \in \mathbb{Z}[X]$ mit $1 = \frac{1}{p}\Phi_p + \frac{1}{p}v \cdot \Phi_1$.

(ii) Es gilt $pL \subseteq L_1 \perp L_p \subseteq L$.

Beweis:

(i) Nach (3.1.4) ist $\Phi_p(X) = \frac{X^p-1}{X-1} = X^{p-1} + X^{p-2} + \cdots + 1$, also ist $\Phi_p(1) = p$ und somit 1 eine Nullstelle von $p - \Phi_p \in \mathbb{Z}[X]$. Da $\Phi_1(X) = X - 1$ folgt daher

$$p - \Phi_p = v \cdot \Phi_1 \text{ für ein } v \in \mathbb{Q}[X]. \tag{4.1}$$

Mit dem Lemma von Gauß muss $v \in \mathbb{Z}[X]$ gelten. Umstellen der Gleichung (4.1) liefert die Behauptung.

(ii) Zu zeigen ist $px \in L_1 \perp L_p$ für alle $x \in L$. Wegen $(\Phi_1\Phi_p)(\sigma) = 0$ und der σ-Invarianz von L ist

$$\begin{aligned} px = \Phi_p(\sigma)(x) + (v \cdot \Phi_1)(\sigma)(x) \in & L \cap \operatorname{Kern}(\Phi_1(\sigma)) + L \cap \operatorname{Kern}(\Phi_p(\sigma)) \\ = & (L \cap V_1) \perp (L \cap V_p) = L_1 \perp L_p. \end{aligned}$$ $\square$

Mit diesem Lemma muss M ein Gitter der Dimension n sein, also gilt $\operatorname{Dim}(L_p) = n_p$ und $\operatorname{Dim}(L_1) = n_1$. Ist L gerade und von quadratfreier Stufe ℓ, so gilt $\ell L^\# \subseteq L$. Lemma (4.2.1)(ii) ist äquivalent zu $pM^\# \subseteq L^\#$. Zusammen erhält man folglich

$$\ell p M^\# \subseteq \ell L^\# \subseteq L.$$

Schneidet man mit V_p, so folgt

$$\ell p(M^\# \cap V_p) \subseteq (L \cap V_p) \Leftrightarrow \ell p L_p^\# \subseteq L_p$$

und analog $\ell p L_1^{\#} \subseteq L_1$.

Im Spezialfall $\mathrm{ggT}(\ell, p) = 1$ bedeutet dies, dass die Stufe der Gitter L_1 und L_p das Produkt ℓp teilt.
Als nächstes wollen wir die Determinanten von L_1 und L_p untersuchen. Dazu werden die partiellen Dualgitter betrachtet.

(4.2.2) Lemma
Sei L ein gerades Gitter der quadratfreien Stufe ℓ und $\sigma \in \mathrm{Aut}(L)$ von Primzahlordnung p mit $\mathrm{ggT}(p, \ell) = 1$.

(i) $pL_1^{\#,p} \subseteq L_1$.

(ii) $(1-\sigma)L_p^{\#,p} \subseteq L_p$.

Beweis:
Teil (i) folgt bereits aus der Definition des partiellen Duals, denn es gilt

$$pL_1^{\#,p} = p\left(\frac{1}{p}L_1 \cap L_1^{\#}\right) = L_1 \cap pL_1^{\#} \subseteq L_1.$$

Kommen wir nun zu Teil (ii). Definiere dazu die Projektionen $\pi_1 := \frac{1}{p}\Phi_p(\sigma)$ und $\pi_p := 1 - \pi_1$ auf V_1 bzw. V_p (vgl. Lemma (4.2.1)). Es zeigt sich:

$$\begin{aligned}
(1-\sigma)\pi_p &= (1-\sigma)(1-\pi_1) \\
&= 1 - \sigma - \pi_1 + \sigma\pi_1 \\
&= 1 - \sigma - \pi_1 + \frac{1}{p}(\sigma^p + \sigma^{p-1} + \cdots + \sigma) \\
&= 1 - \sigma - \pi_1 + \frac{1}{p}(1 + \sigma^{p-1} + \cdots + \sigma) \\
&= 1 - \sigma - \pi_1 + \pi_1 \\
&= 1 - \sigma.
\end{aligned}$$

Sei nun $(b_1, \dots, b_n)$ eine Basis von L mit zugehöriger Dualbasis $(b_1^\#, \dots, b_n^\#)$, sodass $(b_1, \dots, b_{n_p})$ Basis von L_p ist. Dann gilt

$$\pi_p(L^\#) = \pi_p(\langle b_1^\#, \dots, b_n^\# \rangle) = \langle b_1^\#, \dots, b_{n_p}^\# \rangle = L_p^\#.$$

Setzt man diese beiden Fakten zusammen, so erhält man

$$(1-\sigma)L_p^\# = (1-\sigma)\pi_p(L^\#) = (1-\sigma)L^\# \overset{\text{Stufe } \ell}{\subseteq} (1-\sigma)\frac{1}{\ell}L \overset{L\ \sigma\text{-invariant}}{\subseteq} \frac{1}{\ell}L.$$

Außerdem ist $(1-\sigma)L_p^\# \subseteq V_p$, also zusammen

$$(1-\sigma)L_p^\# \subseteq \frac{1}{\ell}L \cap V_p = \frac{1}{\ell}L_p.$$

Für das partielle Dual ergibt sich hiermit

$$(1-\sigma)L_p^{\#,p} = (1-\sigma)\left(\frac{1}{p}L_p \cap L_p^\#\right) \subseteq \frac{1}{p}L_p \cap \frac{1}{\ell}L_p = \frac{1}{\operatorname{ggT}(p,\ell)}L_p = L_p. \qquad \square$$

Wir benötigen noch ein weiteres Hilfslemma.

(4.2.3) Lemma

Sei Λ ein gerades Gitter, dessen Stufe $p\ell$ teilt, wobei p prim und ℓ quadratfrei mit $\operatorname{ggT}(p,\ell) = 1$ sind. Dann ist $\Lambda^{\#,p}/\Lambda \cong \Lambda^\#/\Lambda^{\#,\ell}$.

Beweis:

Sei $\psi : \Lambda^{\#,p} \to \Lambda^\#/\Lambda^{\#,\ell}, x \mapsto x + \Lambda^{\#,\ell}$.

Surjektivität: Sei $x \in \Lambda^\#$. Wegen $p\ell\Lambda^\# \subseteq \Lambda$ ist $p\Lambda^\# \subseteq \Lambda^{\#,\ell}$ und $\ell\Lambda^\# \subseteq \Lambda^{\#,p}$. Nach Euklid existieren Zahlen $s, t \in \mathbb{Z}$ mit $sp + t\ell = 1$. Dann ist $x = spx + t\ell x \subseteq \Lambda^{\#,\ell} + \Lambda^{\#,p}$ und somit $\psi(t\ell x) = x + \Lambda^{\#,\ell}$.

Kern: Der Kern der Abbildung ist $\Lambda^{\#,p} \cap \Lambda^{\#,\ell}$. Es ist einerseits

$$\Lambda^{\#,p} \cap \Lambda^{\#,\ell} \subseteq \frac{1}{p}\Lambda \cap \frac{1}{\ell}\Lambda = \frac{1}{\operatorname{ggT}(p,\ell)}\Lambda = \Lambda$$

und andersherum per Definition $\Lambda \subseteq \Lambda^{\#,p}$ und $\Lambda \subseteq \Lambda^{\#,\ell}$. Insgesamt ist $\mathrm{Kern}(\psi) = \Lambda$.

Die Behauptung folgt nun mit dem Homomorphiesatz. $\square$

Nun wird ein wichtiger Satz zur Bestimmung der Determinanten dargelegt:

(4.2.4) Satz

Sei L wie vorher von quadratfreier Stufe ℓ und $\sigma \in \mathrm{Aut}(L)$ mit $|\sigma| = p$, $\mathrm{ggT}(p, \ell) = 1$. Seien außerdem L_1 und L_p die von σ induzierten Fix- und Bildgitter mit Dimensionen n_1 und n_p. Dann gilt:

$$L_1^{\#,p}/L_1 \cong \mathbb{F}_p^s \cong L_p^{\#,p}/L_p$$

für ein $s \in \{0, \ldots, \min(n_1, \frac{n_p}{p-1})\}$.

Beweis:

Wir zeigen zunächst $L_1^{\#,p}/L_1 \cong L_p^{\#,p}/L_p$. Dies ist nach Lemma (4.2.3) äquivalent zu $L_1^{\#}/L_1^{\#,\ell} \cong L_p^{\#}/L_p^{\#,\ell}$.

Sei $y \in L_1^{\#}$ beliebig. Die Abbildung $L_1 \to \mathbb{Z}, x \mapsto b(x, y)$ ist eine Linearform. Da L_1 der Schnitt von L mit dem Untervektorraum V_1 ist, ist L_1 ein primitiver Teilmodul von L und diese Linearform lässt sich zu einer Linearform auf ganz L fortsetzen. Unter Ausnutzung der Isomorphie $\mathrm{Hom}_{\mathbb{Z}}(L, \mathbb{Z}) \cong L^{\#}$ existiert ein Element $\hat{y} \in L^{\#}$, welches diese Linearform darstellt; insbesondere gilt also $b(x, y) = b(x, \hat{y})$ für alle $x \in L_1$. Zunächst zeigt sich für das Element $\hat{y} - y$:

$$b(x, \hat{y} - y) = 0 \quad \text{für alle } x \in L_1$$

und somit $\hat{y} - y \in V_1^{\perp} = V_p$. Außerdem ist

$$b(x, \hat{y} - y) = b(x, \hat{y}) - b(x, y) = b(x, \hat{y}) \in \mathbb{Z} \quad \text{für alle } x \in L_p.$$

Insgesamt gilt damit $\hat{y} - y \in L_p^{\#}$. Wir können somit die folgende Abbildung definieren:

$$\psi : L_1^{\#} \to L_p^{\#} / L_p^{\#,\ell}, y \mapsto (\hat{y} - y) + L_p^{\#,\ell}.$$

Wir zeigen nun, dass ψ ein wohldefinierter Epimomorphismus mit Kern $L_1^{\#,\ell}$ ist und folgern dann die Behauptung erneut mit dem Homomorphiesatz.

Wohldefiniert: Es definiere $\tilde{y} \in L^{\#}$ eine weitere Fortsetzung. Da L von Stufe ℓ ist, gilt $\hat{y} - \tilde{y} \in L^{\#} \subseteq \frac{1}{\ell} L$. Wir schlussfolgern für $y \in L_1^{\#}$:

$$(\hat{y} - y) - (\tilde{y} - y) = \hat{y} - \tilde{y} \in \frac{1}{\ell} L \cap L_p^{\#} = \frac{1}{\ell} L_p \cap L_p^{\#} = L_p^{\#,\ell}.$$

Das Bild unter ψ hängt daher nicht von der gewählten Fortsetzung ab.

Linearität: Seien $y_1, y_2 \in L_1^{\#}$ mit Elementen $\hat{y_1}, \hat{y_2} \in L^{\#}$, welche die zugehörigen fortgesetzten Linearformen darstellen. Für $s, t \in \mathbb{Z}$ definiert dann $s\hat{y_1} + t\hat{y_2}$ eine Fortsetzung der Linearform $x \mapsto b(x, sy_1 + ty_2)$.

Surjektivität: Sei $y' \in L_p^{\#}$. Es korrespondiere $\hat{y} \in L^{\#}$ zu einer Fortsetzung von $x \mapsto b(x, y) \in \operatorname{Hom}_{\mathbb{Z}}(L_p, \mathbb{Z})$ auf L. Wie zuvor liegt dann das Element $y := \hat{y} - y'$ in $L_1^{\#}$. Durch $\hat{y}$ wird zudem eine Fortsetzung der Linearform $L_1 \to \mathbb{Z}, x \mapsto b(x, y)$ dargestellt, denn für alle $x \in L_1$ ist

$$b(x, y) = b(x, \hat{y} - y') = b(x, \hat{y}) - b(x, y') = b(x, \hat{y}).$$

Somit ist $\psi(y) = (\hat{y} - y) + L_1^{\#,\ell} = y' + L_1^{\#,\ell}$.

Kern: Es ist $\operatorname{Kern}(\psi) \subseteq L_1^{\#,\ell}$, denn sei $y \in \operatorname{Kern}(\psi)$, so gilt $\hat{y} - y \in \frac{1}{\ell} L_p^{\#,\ell} \subseteq \frac{1}{\ell} L_p \subseteq \frac{1}{\ell} L$. Da zudem $\hat{y} \in L^{\#} \subseteq \frac{1}{\ell} L$ ist, folgt $y = \hat{y} - (\hat{y} - y) \in \frac{1}{\ell}$. Insgesamt gilt daher $y \in \frac{1}{\ell} L \cap L_1^{\#} = \frac{1}{\ell} L_1 \cap L_1^{\#} = L_1^{\#,\ell}$.

Andersherum ist $L_1^{\#,\ell} \subseteq \operatorname{Kern}(\psi)$, denn sei $y \in L_1^{\#,\ell}$, so ist $y \in \frac{1}{\ell} L_1 \subseteq \frac{1}{\ell} L$. Somit gilt $\hat{y} - y \in \frac{1}{\ell} L \cap L_p^{\#} = \frac{1}{\ell} L_p \cap L_p^{\#} = L_p^{\#,\ell}$ und damit $y \in \operatorname{Kern}(\psi)$.

Die erste Behauptung folgt nun aus dem Homomorphiesatz.

Verwendet man nun Lemma (4.2.2), so zeigt sich, dass $L_1^{\#,p}/L_1$ ein Quotient der Gruppe $L_1^{\#,p}/pL_1^{\#,p} \cong \mathbb{F}_p^{n_1}$ ist und somit die Gestalt $\mathbb{F}_p^s$ für ein $s \in \{0, \ldots, n_1\}$ besitzt.
Analog zeigt dasselbe Lemma, dass $L_p^{\#,p}/L_p$ ein Faktor der Gruppe $L_p^{\#,p}/(1-\sigma)L_p^{\#,p} \cong (\mathbb{Z}[\zeta_p]/(1-\zeta_p)\mathbb{Z}[\zeta_p])^{\frac{n_p}{p-1}}$ ist. Hier ist

$$\begin{aligned} p + (1-\zeta_p)\mathbb{Z}[\zeta_p] &= \underbrace{1 + \cdots + 1}_{p \text{ mal}} + (1-\zeta_p)\mathbb{Z}[\zeta_p] \\ &= 1^{p-1} + \cdots + 1^0 + (1-\zeta_p)\mathbb{Z}[\zeta_p] \\ &= \zeta_p^{p-1} + \cdots + \zeta_p^0 + (1-\zeta_p)\mathbb{Z}[\zeta_p] \\ &= 0 + (1-\zeta_p)\mathbb{Z}[\zeta_p], \end{aligned}$$

diese enthält also genau p Elemente und wir erhalten finalerweise, dass $L_p^{\#,p}/L_p$ ein Quotient von $(\mathbb{F}_p)^{\frac{n_p}{p-1}}$ ist. Damit folgt $s \leq \frac{n_p}{p-1}$. □

Wir können entsprechend die Faktorgruppen $L_1^{\#,p}/L_1$ und $L_p^{\#,p}/L_p$ als $\mathbb{F}_p$-Vektorräume der Dimension s auffassen.

Nach Lemma (4.2.3) ist

$$\operatorname{Det}(L_p) = \left[L_p^{\#} : L_p\right] = \left[L_p^{\#} : L_p^{\#,\ell}\right] \cdot \left[L_p^{\#,\ell} : L_p\right] = \left[L_p^{\#,p} : L_p\right] \cdot \left[L_p^{\#,\ell} : L_p\right].$$

Außerdem teilt p den Index $\left[L_p^{\#,\ell} : L_p\right]$ nicht, da der Exponent der Faktorgruppe $L_p^{\#,\ell}/L_p$ wegen $\ell L_p^{\#,\ell} \subseteq L_p$ ein Teiler von ℓ sein muss und $\operatorname{ggT}(\ell, p) = 1$. Ist also s wie im vorigen Satz, so ist s bereits die p-Bewertung der Determinante von L_p. Die Überlegungen funktionieren selbstverständlich analog für L_1. Der Satz sagt uns also, dass $\operatorname{Det}(L_1) = p^s c$ und $\operatorname{Det}(L_p) = p^s d$ für gewisse, zu p teilerfremde, $c, d \in \mathbb{N}$. Nach Lemma (4.2.1) ist der Index $[L : M]$ allerdings eine p-Potenz, während die Determinante von L teilerfremd zu p ist. Daher muss $c \cdot d = \operatorname{Det}(L)$ gelten und sich der in Abbildung (4.1) dargestellte Inklusionsverband ergeben.

$$
\begin{tikzcd}
L_1^{\#} \perp L_p^{\#} \arrow[d, dash, "p^s"] \\
L^{\#} \arrow[d, dash, "\mathrm{Det}(L)"] \\
L \arrow[d, dash, "p^s"] \\
L_1 \perp L_p
\end{tikzcd}
$$

Abbildung 4.1: Inklusionsverband

Diese Überlegungen erlauben uns folgende Definition:

(4.2.5) Definition

Sei L ein gerades Gitter der quadratfreien Stufe ℓ. Sei weiterhin $\sigma \in \mathrm{Aut}(L)$ von Ordnung p und L_1 und L_p mit Dimensionen n_1 und n_p wie zuvor die von σ induzierten Teilgitter. Die Primfaktorzerlegung von ℓ sei gegeben durch $\ell = q_1 \dots q_m$. Ist $\mathrm{Det}(L_1) = p^s q_1^{k_{1,1}} \dots q_m^{k_{1,m}}$ und $\mathrm{Det}(L_p) = p^s q_1^{k_{p,1}} \dots q_m^{k_{p,m}}$, so nennen wir das Tupel

$$p - (n_1, n_p) - s - q_1 - (k_{1,1}, k_{p,1}) - q_2 - (k_{1,2}, k_{p,2}) - \dots - q_m - (k_{1,m}, k_{p,m})$$

den *Typen* von σ.

Ist $m = 1$, also ℓ prim, so können wir die Schreibweise verkürzen zu

$$p - (n_1, n_p) - s - (k_1, k_p).$$

Wir können nun einige Einschränkungen an den Typen eines solchen Automorphismus machen. Im Spezialfall $p = 2$ hat die Gruppe $M^{\#,2}/M$ Exponent 2, in diesem Fall gilt eine veränderte Version von [Neb13, Lemma (4.9)].

(4.2.6) Lemma

Sei M ein gerades Gitter in einem bilinearen Vektorraum (V, b) und $M^{\#,2}/M$ habe Exponent 2. Dann enthält M ein Teilgitter isometrisch zu $\sqrt{2}U$, wobei U ein Gitter mit $U = U^{\#,2}$ ist und der Index $\left[M : \sqrt{2}U\right]$ eine Zweierpotenz.

Beweis:

Wir betrachten die 2-adische Jordanzerlegung (vgl. [CS93, (7.1)])

$$\mathbb{Z}_2 \otimes M \cong f_1 \perp \sqrt{2}f_2$$

mit einem geraden Gitter f_1 und einem ganzen Gitter f_2, sodass $\mathrm{Det}(f_1)$ und $\mathrm{Det}(f_2)$ teilerfremd zu 2 sind. Da f_1 ein regulärer $\mathbb{Z}_2$-Modul ist, erlaubt uns [Kne02, Satz (4.1)] eine Zerlegung

$$f_1 = E_1 \perp E_2 \perp \cdots \perp E_m$$

mit regulären Teilmoduln E_i von Dimension ≤ 2. Da jedes E_i ein gerades Gitter sein muss, folgt $\mathrm{Dim}(E_i) = 2$ für alle $i = 1, \ldots, m$. Insbesondere hat f_1 die gerade Dimension $2m$.

Ist $m = 0$, so sind wir fertig mit $U := f_2$. Sei also nun $m > 0$.
Wir zeigen nun, f_1 enthält ein Element v mit $b(v, v) \in 2\mathbb{Z}_2^*$. Dazu schreiben wir $E_1 = \langle x, y\rangle$ und definieren s, t durch $b(x, x) \in 2^s\mathbb{Z}_2^*$ und $b(y, y) \in 2^t\mathbb{Z}_2^*$. Ist $s = 1$ oder $t = 1$, so haben wir mit $v := x$ bzw. $v := y$ ein solches Element gefunden. Seien also nun $s, t \geq 2$. Dann folgt

$$b(x - y, x - y) = b(x, x) + b(y, y) - 2b(x, y).$$

Nun muss $b(x, y)$ in $\mathbb{Z}_2^*$ liegen, da sonst die Gram-Matrix von E_1 nur gerade Einträge und somit auch eine gerade Determinante hätte. Somit erhalten wir

$$b(x - y, x - y) \in 2^s\mathbb{Z}_2^* + 2^t\mathbb{Z}_2^* + 2\mathbb{Z}_2^* = 2(\underbrace{2^{s-1}\mathbb{Z}_2^* + 2^{t-1}\mathbb{Z}_2^*}_{\subseteq 2\mathbb{Z}_2} + \mathbb{Z}_2^*) \subseteq 2\mathbb{Z}_2^*.$$

Damit ist $v := x - y$ ein Element wie gesucht.

Nach den obrigen Überlegungen können ohne Einschränkung annehmen, dass $E_1 = \langle x, v \rangle$, sonst vertausche x und y. Setze nun $w := b(v,v)x - b(v,x)v$, dann ist $b(v,w) = b(v,v)b(v,x) - b(v,x)b(v,v) = 0$ und mit $b(x,v) \in \mathbb{Z}_2^*$ außerdem

$$b(w,w) = b(v,v)^2 b(x,x) - b(v,v)b(x,v)^2 \in 2^{2+s}\mathbb{Z}_2^* + 2\mathbb{Z}_2^* = 2\mathbb{Z}_2^*.$$

Nun folgt: $\langle v \rangle \perp \langle w \rangle \perp E_2 \perp \cdots \perp E_m$ ist ein Teilgitter von f_1 vom Index 2 und $\langle v \rangle \perp \langle w \rangle = \sqrt{2}g$ für ein reguläres, ganzes Gitter g. Insgesamt ist somit

$$U' := \langle v \rangle \perp \langle w \rangle \perp E_2 \perp \cdots \perp E_m \perp \sqrt{2}f_2$$

ein Teilgitter von M vom Index 2 und mit Jordanzerlegung $(E_2 \perp \cdots \perp E_m) \perp \sqrt{2}\,(g \perp f_2)$. Eine Induktion liefert nun die Behauptung. $\square$

Ist M wie im obigen Lemma mit Determinante $2^s a$ für ein ungerades $a \in \mathbb{N}$, dann hat das Gitter U Determinante a und Minimum $\mathrm{Min}(U) \geq \frac{\mathrm{Min}(M)}{2}$. Nun können wir einige Einschränkungen an die Typ-Parameter zeigen.

(4.2.7) Satz

Sei L ein gerades, n-dimensionales Gitter der quadratfreien Stufe ℓ mit Determinante $\mathrm{Det}(L) = \ell^k$. Sei zudem $\sigma \in \mathrm{Aut}(L)$ von Typ $p - (n_1, n_p) - s - q_1 - (k_{1,1}, k_{p,1}) - \cdots - q_m - (k_{1,m}, k_{p,m})$, wobei $\mathrm{ggT}(p, \ell) = 1$. Dann gelten folgende Einschränkungen (für alle $i \in \underline{m}$):

(i) $n_1 + n_p = n$.

(ii) $s \in \{0, \ldots, \min(n_1, \frac{n_p}{p-1})\}$.

(iii) $s \equiv_2 \frac{n_p}{p-1}$ und für $p = 2$ zusätzlich $s \equiv_2 0$.

(iv) $k_{1,i} \in \{0, \ldots, \min(n_1, k)\}$.

(v) $k_{1,i} \equiv_2 k$.

(vi) $k_{p,i} \in \{0, \ldots, \min(n_p, k)\}$.

(vii) $k_{p,i} \equiv_2 0$.

(viii) $(2f(q_i))\,|k_{p,i}$, wobei $f(q_i)$ den Trägheitsgrad von $q_i\mathbb{Z}_{\mathbb{Q}(\zeta_p+\zeta_p^{-1})}$ bezeichne.

(ix) $k_{1,i} + k_{p,i} = k$.

Beweis:

Eigenschaft (i) ist klar, (ii) ist Satz (4.2.4), Nummer (iv), (vi) und (ix) ergeben sich daraus, dass die Stufen von L_p und L_1 Teiler von $p\ell$ sind und der Tatsache, dass $\frac{\mathrm{Det}(L_1)\mathrm{Det}(L_p)}{p^{2s}} = \mathrm{Det}(L)$.

Nach [Jü15, Satz (3.1.4)(d) und Lemma (3.1.1)] existiert ein $\mathbb{Z}_{\mathbb{Q}(\zeta_p+\zeta_p^{-1})}$-Ideal $\mathfrak{a}$ mit

$$\mathrm{Det}(L_p) = p^s q_1^{k_{p,1}} \ldots q_m^{k_{p,m}} = p^{(p-2)\frac{n_p}{p-1}} \cdot \mathcal{N}(\mathfrak{a})^2.$$

Mit $\mathrm{ggT}(p, \ell) = 1$ zeigt die Gleichung sofort Eigenschaft (vii). Außerdem kann man ablesen, dass $s \equiv_2 (p-2)\frac{n_p}{p-1}$ sein muss. Für (iii) bleibt also lediglich zu zeigen, dass im Falle $p = 2$ ebenfalls $s \equiv_2 \frac{n_p}{p-1}$ gilt. Hierfür verwenden wir Lemma (4.2.6) mit $M := L_p$. Für das U aus dem Lemma gilt

$$2^{n_p} = |(\sqrt{2}U)^{\#,2}/\sqrt{2}U| = |L_2^{\#,2}/L_2|[L_2 : U]^2 = 2^{s+2\nu_2([L_2:U])}.$$

Zuletzt ergibt sich (v) aus (vii) und (ix). Teil (viii) ist [Jü15, Korollar (4.1.9)]. $\square$

Wir können nun mithilfe der Einschränkungen aus Satz (4.2.7) und den Hermite-Schranken aus (2.2) den Algorithmus (5) entwerfen, welcher die möglichen Automorphismentypen gerader Gitter mit quadratfreier Stufe zurückgibt.

Algorithmus 5 Aufzählung von Automorphismen-Typen

Eingabe: Quadratfreies $\ell \in \mathbb{N}$, $k \in \mathbb{N}$, $n \in \mathbb{N}$, $t \in \mathbb{N}$.

Ausgabe: Liste aller Typen von Automorphismen mit Primzahlordnung p von geraden Gittern der Stufe ℓ, Determinante ℓ^k, Dimension n, und Minimum $\geq t$, wobei $\mathrm{ggT}(p, \ell) = 1$.

Res $\leftarrow [\,]$

$b \leftarrow$ Liste von Schranken für die Hermite-Konstante γ_i für $1 \leq i \leq n$

$q_1, \ldots, q_m \leftarrow$ Primfaktoren von ℓ

for $p \in \mathbb{P}_{\leq n+1} - \{q_1, \ldots, q_m\}$ **do**

  $f_i :=$ Trägheitsgrad von $q_i \mathbb{Z}_{\mathbb{Q}(\zeta_p + \zeta_p^{-1})}$ für $i = 1, \ldots, m$

  for $n_p \in \{i(p-1) \mid 1 \leq i \leq \lfloor \frac{n}{p-1} \rfloor\}$ **do**

    $n_1 \leftarrow n - n_p$

    for $(k_{p,1}, k_{p,2}, \ldots, k_{p,m}) \in \prod_{i=1}^{m} \{(2f_i)j \mid j \in \{0, \ldots, \lfloor \frac{\min(n_p,k)}{2f_i} \rfloor\}\}$ **do**

      $k_{1,i} \leftarrow k - k_{p,i}, \quad i \in \underline{m}$

      if $\exists i \in \underline{m} : (k_{1,i} > \min(n_1, k)) \vee (k_{1,i} \not\equiv_2 k) \vee (k_{p,i} \not\equiv_2 0)$ **then**

        continue

      for $s \in \{0, \ldots, \min(n_1, \frac{n_p}{p-1})\}$ **do**

        if $s \not\equiv_2 (p-2)\frac{n_p}{p-1}$ **then continue**

        $\gamma_1 \leftarrow \frac{t}{\left(p^s q_1^{k_{1,1}} \ldots q_m^{k_{1,m}}\right)^{1/n_1}}$

        $\gamma_p \leftarrow \frac{t}{\left(p^s q_1^{k_{p,1}} \ldots q_m^{k_{p,m}}\right)^{1/n_p}}$

        if $\gamma_1 > b_{n_1}$ oder $\gamma_p > b_{n_p}$ **then continue**

        if $p = 2$ **then**

          $\gamma_1' \leftarrow \frac{t/2}{\left(q_1^{k_{1,1}} \ldots q_m^{k_{1,m}}\right)^{1/n_1}}$

          $\gamma_p' \leftarrow \frac{t/2}{\left(q_1^{k_{p,1}} \ldots q_m^{k_{p,m}}\right)^{1/n_p}}$

          if $\gamma_1' > b_{n_1}$ oder $\gamma_p' > b_{n_p}$ **then continue**

        Res $\leftarrow$ Res $\cup\, [p - (n_1, n_p) - s - q_1 - (k_{1,1}, k_{p,1}) - \cdots - (k_{1,m}, k_{p,m})]$

return Res

Wir haben in diesem Kapitel die möglichen Typen von Automorphismen mit Primzahlordnung studiert und Aussagen über die Determinanten der induzierten Teilgitter getroffen. Als nächstes definieren wir den Begriff des *Geschlechts* von Gittern und beschreiben eine Möglichkeit zur Aufzählung eines Geschlechts

§ 4.3 Geschlechter

(4.3.1) Definition

Wir sagen, zwei ganze Gitter L und L' liegen im selben *Geschlecht*, wenn

$$\mathbb{R} \otimes L \cong \mathbb{R} \otimes L' \quad \text{und} \quad \mathbb{Z}_p \otimes L \cong \mathbb{Z}_p \otimes L' \text{ für alle } p \in \mathbb{P}.$$

Dabei bezeichnet $\mathbb{Z}_p$ den Ring der p-adischen ganzen Zahlen.

Conway und Sloane geben in[CS93, Kap. 15, Abs. 7] eine trennende Invariante der Geschlechter von Gittern an, das sogenannte *Geschlechtssymbol*. Dessen Gestalt und Eigenschaften werden im Folgenden erläutert.

Das gesamte Symbol setzt sich aus mehreren lokalen Symbolen zusammen; eines für jede Primzahl und eines für (-1).

Sei L ein ganzes Gitter in einem bilinearen $\mathbb{Q}$-Vektorraum (V, b). Das (-1)-adische Symbol ist von der Gestalt

$$+^r -^s$$

und beschreibt die Signatur der Bilinearform b.

Für die p-adischen Symbole, wobei $p \in \mathbb{P}$, betrachte die Jordanzerlegung

$$\mathbb{Z}_p \otimes L \cong f_1 \perp \sqrt{p} f_p \perp \cdots \perp \sqrt{p}^k f_{p^k}.$$

Klar ist: Haben zwei Gitter die gleiche Signatur und über allen Primzahlen die gleiche Jordanzerlegung, so liegen sie im gleichen Geschlecht. Die Jordanzerlegung ist im allgemeinen nicht eindeutig, es ist jedoch bekannt, inwiefern sich zwei unterschiedliche Jordanzerlegungen desselben Gitters unterscheiden.
Für $p > 2$ ist die Zerlegung bis auf die Determinanten der Teilgitter $f_1, \dots f_{p^k}$ eindeutig, welche sich um ein Quadrat unterscheiden können. Genauer: Definiere die Invarianten

$$n_q := \dim(f_q), \quad \epsilon_q := \left(\frac{\operatorname{Det}(f_q)}{p}\right) := \begin{cases} +1 & , \ \operatorname{Det}(f_q) \in (\mathbb{Z}_p^*)^2 \\ -1 & , \ \operatorname{Det}(f_q) \notin (\mathbb{Z}_p^*)^2 \end{cases}$$

für $q \in \{1, p, \dots, p^k\}$, dann sind zwei Gitter genau dann isometrisch über $\mathbb{Z}_p$, wenn sie dieselben Invarianten n_q und ϵ_q haben. Wir definieren nun das p-adische Symbol für $p > 2$ als das formale Produkt

$$1^{\epsilon_1 n_1} \ p^{\epsilon_p n_p} \dots \left(p^k\right)^{\epsilon_{p^k} n_{p^k}} .$$

Beispielsweise bedeutet das Symbol $1^{-2}3^{+5}$, dass jede Jordanzerlegung des Gitters über $\mathbb{Z}_3$ die Form $\mathbb{Z}_3 \otimes L \cong f_1 \perp \sqrt{3} f_3$ besitzt mit $\dim(f_1) = 2$, $\dim(f_3) = 5$. Außerdem ist $\operatorname{Det}(f_1)$ kein Quadrat, aber $\operatorname{Det}(f_3)$ ein Quadrat mod 3.

Der Fall $p = 2$ ist aufwändiger. Da b ursprünglich eine Bilinearform über $\mathbb{Q}$ ist, können wir sie nach [Kne02, Satz (1.20)] diagonalisieren. Sei G_q jeweils die diagonalisierte Matrix der Bilinearform auf dem zu f_q gehörigen Teilraum. Wir definieren

$$\begin{aligned} n_q &:= \dim(f_i) \\ S_q &:= \begin{cases} \mathrm{I} & , \ f_q \text{ ist kein gerades Gitter} \\ \mathrm{II} & , \ f_q \text{ ist gerades Gitter} \end{cases} \\ \epsilon_q &:= \left(\frac{\operatorname{Det}(f_q)}{2}\right) := \begin{cases} +1 & , \ \operatorname{Det}(f_q) \equiv_8 \pm 1 \\ -1 & , \ \operatorname{Det}(f_q) \equiv_8 \pm 3 \end{cases} \end{aligned}$$

$$t_q := \begin{cases} \mathrm{Spur}(G_q) \bmod 8 & , \; S_q = \mathrm{I} \\ 0 & , \; S_q = \mathrm{II} \end{cases}$$

für $q \in \{1, 2, 4, \ldots, 2^k\}$. Wir erhalten nun das vorläufige Symbol

$$1^{\epsilon_1 n_1}_{t_1/\mathrm{II}} \; 2^{\epsilon_2 n_2}_{t_2/\mathrm{II}} \cdots \left(2^k\right)^{\epsilon_{2^k} n_{2^k}}_{t_{2^k}/\mathrm{II}},$$

Wwbei der Index für Komponenten mit $S_q = \mathrm{I}$ den Wert t_q darstellt und andernfalls II ist. Dieses Symbol ist noch immer nicht eindeutig, wir benötigen zusätzliche Normierungsbedingungen. Wir fassen nun maximale Teilintervalle mit der Eigenschaft, dass alle Faktoren in den Intervallen den Typ $S_q = \mathrm{I}$ haben (mit eckigen Klammern) zu sogenannten *Abteilen* zusammen. Außerdem trennen wir (mit Doppelpunkten) das Symbol in maximale Teilintervalle, genannt *Züge*, sodass in jedem Zug mindestens einer von je zwei aufeinanderfolgenden Faktoren den Typ $S_q = \mathrm{I}$ hat. Beispielsweise wird so das Symbol

$$1^{+2}_{\mathrm{II}} \; 2^{-2}_{6} \; 4^{+3}_{5} \; 8^{+0}_{\mathrm{II}} \; 16^{+1}_{\mathrm{II}} \; 32^{+2}_{\mathrm{II}} \; 64^{+1}_{3} \tag{4.2}$$

zu

$$1^{+2}_{\mathrm{II}} \left[2^{-2}_{6} \; 4^{+3}_{5}\right] 8^{+0}_{\mathrm{II}} : 16^{+1}_{\mathrm{II}} : 32^{+2}_{\mathrm{II}} \left[64^{+1}_{3}\right].$$

Es gibt nun zwei mögliche Transformationen des Symbols, sodass diese Jordanzerlegungen desselben Gitters entsprechen.

Erstens stellen zwei solche Symbole das gleiche Gitter dar, wenn sie sich nur durch Änderungen der t_q bei den Typ-I-Faktoren unterscheiden, die Summen aller t_q pro Abteil jedoch kongruent sind modulo 8. Nach dieser Regel genügt es also, im 2-adischen Symbol jeweils nur einen Index pro Abteil anzugeben, der die Summe der enthaltenen t_q modulo 8 darstellt.

Zweitens sind zwei Symbole äquivalent, die durch beliebig häufige Anwendung der folgenden Schritte auseinander hervorgehen:

- Wähle $2^{k_1} < 2^{k_2} \in \{1, 2, \ldots, 2^k\}$ so, dass die zugehörigen Komponenten im selben Zug liegen.

- Setzte $\epsilon_{2^{k_1}} = -\epsilon_{2^{k_1}}$ und $\epsilon_{2^{k_2}} = -\epsilon_{2^{k_2}}$.
- Definiere den Weg $W := \{(a, 2a) \mid a = k_1, 2k_1, \ldots, \frac{1}{2}k_2\}$. In jedem Abteil, sodass $|\{(a, 2a) \in W \mid a \text{ oder } 2a \text{ im Abteil}\}|$ ungerade, ändere die Werte t_q aller Komponenten so, dass sich die Summe dieser um genau 4 modulo 8 unterscheidet.

Beispielsweise korrespondieren die Symbole

$$1_{\mathrm{II}}^{+2}\left[2^{-2}\ 4^{+3}\right]_3 8_{\mathrm{II}}^{+0}\left[16^{-1}\right]_5$$

und

$$1_{\mathrm{II}}^{+2}\left[2^{+2}\ 4^{+3}\right]_3 8_{\mathrm{II}}^{+0}\left[16^{+1}\right]_1$$

zum gleichen Gitter. Es wurden die Vorzeichen bei 2 und 16 verändert. Dabei involvieren zwei Schritte das erste Abteil und ein Schritt das zweite Abteil, also muss der Index des zweiten Abteils um 4 geändert werden, der Index des ersten Abteils jedoch nicht.
Als Normierungsbedingung für diese Transformation können wir fordern, dass jeder Zug höchstens einmal das Vorzeichen $\epsilon_q = -1$ enthalten soll und dass dieses bei der ersten Komponente von Dimension $n_q > 0$ auftritt. Dies schließt die Beschreibung des 2-adischen Symbols ab. Es können nur noch Änderungen vorgenommen werden, die zur besseren Lesbarkeit beitragen, wie etwa das Auslassen der Komponenten von Dimension 0 und der II-Indizes. So hat beispielsweise das fertige 2-adische Symbol von (4.2) die Form

$$1^{-2}\left[2^{+2}4^{+3}\right]_7 : 16^{+1} : 32^{+2}\left[64^{+1}\right]_3.$$

Als nächstes stellt sich andersherum die Frage, zu welchen möglichen Symbolen überhaupt Gitter existieren können. Dazu müssen folgende Bedingungen erfüllt sein.

(i) Für alle $p \in \mathbb{P}$ gilt $\prod_{q \in \{1, p, p^2, \ldots\}} \epsilon_q = \left(\frac{a}{p}\right)$, wobei $\mathrm{Det}(L) = p^\alpha a$ und $\mathrm{ggT}(p, a) = 1$.

(ii) Sei für $p \in \mathbb{P}$ der Wert k_p definiert als die Anzahl der Potenzen q von p, sodass q kein Quadrat ist, aber $\epsilon_q = -1$. Dann ist

$$r - s + \sum_{p>2} \left(4k_p + \sum_{q \in \{1,p,p^2,\dots\}} n_q(q-1) \right) \equiv 4k_2 + \sum_{q \in \{1,2,4,\dots\}} t_q \pmod 8.$$

(iii) Für alle q mit $n_q = 0$ gilt $\epsilon_q = +1$.

(iv) Sei q eine Zweierpotenz. Dann gilt:

- $n_q = 0 \Rightarrow S_q = \mathrm{II}$ und $\epsilon_q = +1$.
- $n_q = 1, \epsilon_q = +1 \Rightarrow t_q \equiv_8 \pm 1$.
- $n_q = 1, \epsilon_q = -1 \Rightarrow t_q \equiv_8 \pm 3$.
- $n_q = 2, S_q = \mathrm{I}, \epsilon_q = +1 \Rightarrow t_q \equiv_8 0$ oder ± 2.
- $n_q = 2, S_q = \mathrm{I}, \epsilon_q = -1 \Rightarrow t_q \equiv_8 4$ oder ± 2.
- $n_q \equiv_2 t_q$.
- n_q ungerade $\Rightarrow S_q = \mathrm{I}$.

Erfüllt ein System von p-adischen Symbolen für $p \in \mathbb{P} \cup \{-1\}$ alle genannten Bedingungen, dann exisitiert ein ganzes Gitter mit diesen Symbolen. Die Gleichung (ii) bezeichnen Conway und Sloane auch als *Oddity-Formel*.

Wir können nun alle lokalen Symbole zum gesamten Geschlechtssymbol kombinieren. Dieses hat die Form

$$\mathrm{I}_{r-s}(\dots), \quad \text{bzw.} \quad \mathrm{II}_{r-s}(\dots).$$

Wobei I/II dem Typen S_1 entspricht, r, s der reellen Signatur und die Klammern die p-adischen Symbole für $p \in \mathbb{P}$ enthalten. Dabei werden die Komponenten zur Potenz

0 jeweils ausgelassen. Deren Invarianten lassen sich jedoch mithilfe der Dimension und Determinante von L sowie den angegebenen Bedingungen an die p-adischen Symbole bei Bedarf herleiten.

(4.3.2) Beispiele

(i) Das Gitter $L := A_2 \times D_4$ hat Dimension 8 und Determinante $2^4 3^4$. Über $\mathbb{Z}_2$ hat es eine Jordanzerlegung

$$\mathbb{Z}_2 \otimes L \cong f_1 \perp \sqrt{2} f_2,$$

wobei f_1 und f_2 gerade Gitter sind, $\dim(f_2) = 4$ und $\mathrm{Det}(f_2) = 225 \equiv_8 1$. Über $\mathbb{Z}_3$ hat es eine Zerlegung

$$\mathbb{Z}_3 \otimes L \cong g_1 \perp \sqrt{3} g_2,$$

wobei $\dim(g_2) = 4$ und $\mathrm{Det}(g_2) = 1$, also ein Quadrat in $\mathbb{Z}_3^*$, ist. Das Geschlecht von L hat somit das Geschlechtssymbol

$$\mathrm{II}_8(2^{+4}\ 3^{+4}).$$

(ii) Ist L ein positiv-definites, gerades, n-dimensionales Gitter der Stufe $\ell \in \mathbb{P}_{>2}$ mit Determinante $\ell^{\frac{n}{2}}$, so ergibt sich aus der Oddity-Formel die Gleichung

$$\frac{n(\ell+1)}{2} \equiv_8 4k_\ell.$$

Weiterhin muss $\epsilon_1 \epsilon_3 = \left(\frac{\mathrm{Det}(L)/\ell^{\frac{n}{2}}}{\ell}\right) = 1$ oder äquivalent $\epsilon_1 = \epsilon_\ell$ sein. Da genau dann $k_\ell = 1$ gilt, wenn $\epsilon_\ell = 1$ und sonst $k_\ell = 0$ ist, ist das zu L gehörige Geschlechtssymbol:

$$\begin{aligned} &\mathrm{II}_n\left(\ell^{+\frac{n}{2}}\right), \quad \text{falls } \frac{n(\ell+1)}{2} \equiv_8 0, \\ &\mathrm{II}_n\left(\ell^{-\frac{n}{2}}\right), \quad \text{falls } \frac{n(\ell+1)}{2} \equiv_8 4. \end{aligned}$$

(iii) Ähnlich können wir für 2-modulare Gitter vorgehen. Sei L ein positiv-definites, gerades, n-dimensionales Gitter der Stufe 2 mit Determinante $2^{\frac{n}{2}}$. Wie zuvor muss $\epsilon_1 = \epsilon_2$ sein und $k_2 = 1 \Leftrightarrow \epsilon_2 = 1$. Die Oddity-Formel ergibt

$$n \equiv_8 4k_2 + t_2.$$

Weiterhin können wir die Modularität von L ausnutzen. Sei

$$\mathbb{Z}_2 \otimes L \cong f_1 \perp \sqrt{2} f_2$$

die 2-adische Jordanzerlegung von L, dann hat $L^{\#}$ die Jordanzerlegung

$$\mathbb{Z}_2 \otimes L^{\#} = f_1 \perp \sqrt{2}^{-1} f_2$$

und somit

$$\mathbb{Z}_2 \otimes (\sqrt{2} L^{\#}) = f_2 \perp \sqrt{2} f_1.$$

Nun ist aber $\sqrt{2} L^{\#} \cong L$ und damit ein gerades Gitter, also muss auch f_2 gerade sein und $t_2 = 0$. Erneut ist allein anhand der Dimension das Geschlechtssymbol eindeutig festgelegt:

$$\begin{aligned} &\mathrm{II}_n\left(2^{+\frac{n}{2}}\right), \quad \text{falls } n \equiv_8 0, \\ &\mathrm{II}_n\left(2^{-\frac{n}{2}}\right), \quad \text{falls } n \equiv_8 4, \end{aligned}$$

genau wie bei ℓ-modularen Gittern für $\ell \in \mathbb{P}_{>2}$.

Jürgens beschreibt in [Jü15, Abschnitt (4.1.3)], welche Gestalt die Geschlechtssymbole der von einem Automorphismus von Primzahlordnung induzierten Fix- und Bildgitter L_1 und L_p besitzen.

(4.3.3) Satz

Sei L ein Gitter der primen Stufe $\ell \in \mathbb{P}$ mit einem Automorphismus σ von Typ $p-(n_1,n_p)-s-(k_1,k_p)$ für ein $p \in \mathbb{P}_{>2}$, wobei $\mathrm{ggT}(p,\ell) = 1$. Wie zuvor seien außerdem $L_1 = L \cap \mathrm{Kern}(\sigma-1)$ und $L_p = L \cap \mathrm{Bild}(\sigma-1)$. Die Geschlechter von L, L_1 und L_p haben die Formen

$$L \in \mathrm{II}_n(\ell^{\epsilon k}), \quad L_1 \in \mathrm{II}_{n_1}(p^{\delta_1 s}\ell^{\epsilon_1 k_1}), \quad L_p \in \mathrm{II}_{n_p}(p^{\delta_p s}\ell^{\epsilon_p k_p})$$

und für die Parameter $\epsilon, \delta_1, \epsilon_1, \delta_p, \epsilon_p$ gelten die folgenden Beziehungen:

(i) $\epsilon_1 \epsilon_p = \epsilon$

(ii) $\delta_1 \delta_p = (-1)^{\frac{s(p-1)}{2}}$

(iii) $\delta_p = (-1)^{\frac{k_p}{f(\ell)} + \frac{p-1}{2}\left(\binom{n_p/(p-1)+1}{2} + \binom{s}{2}\right)}$

(iv) falls $\ell \neq 2$: $\epsilon_p = (-1)^{\frac{k_p}{f(\ell)} + \frac{l-1}{2}\binom{k_p}{2}}$

(v) falls $\ell = 2$: $\epsilon_p = \delta_p \Leftrightarrow n_p + s(p-1) \equiv_8 0$.

Dabei bezeichnet $f(\ell)$ den Trägheitsgrad von $\ell\mathbb{Z}_{\mathbb{Q}(\zeta_p+\zeta_p^{-1})}$.

Die Geschlechtssymbole sind unter den gegebenen Voraussetzungen also eindeutig durch den Typen des Automorphismus festgelegt.

§ 4.4 Kneser-Nachbarschaftsmethode

Wir kommen nun zur Aufzählung aller Gitter eines gegebenen Geschlechtes. Kneser beschreibt in [Kne02, Abschnitt 28] eine Methode, welche Gitter in *Spinorgeschlechtern* mithilfe einer Nachbar-Konstruktion aufzählt.

(4.4.1) Definition
Sei p eine Primzahl sowie L und M $\mathbb{Z}$-Gitter im gleichen Vektorraum. Man bezeichnet L und M als *p-Nachbarn*, falls $[L : L \cap M] = p = [M : L \cap M]$.

Sei $\mathcal{G}$ ein Geschlecht und C die Menge aller Isometrie-Klassen von Gittern in $\mathcal{G}$. Der Graph mit Knotenmenge C und Kanten zwischen $C_1, C_2 \in C$ genau dann, wenn C_1 und C_2 p-Nachbarn für eine Primzahl p, heißt *Nachbarschafts-Graph*.

Kneser beschreibt, wie die Menge aller p-Nachbarn eines gegebenen Gittes L gebildet werden kann. Nach [SH98] ist der Nachbarschafts-Graph endlich. Außerdem gilt der folgende Satz:

(4.4.2) Satz
Sei L ein Gitter von Dimension ≥ 3. Hat für jede Primzahl $q \in \mathbb{P}$ die Jordanzerlegung von $\mathbb{Z}_q \otimes L$ mindestens eine Komponente von Dimension ≥ 2, so besteht jeder Nachbarschafts-Graph von L aus genau einer Zusammenhangskomponente.

Die recht schwache Bedingung zur Jordanzerlegung ist beispielsweise für alle Gitter von quadratfreier Stufe - also für alle von uns zu untersuchenden Geschlechter - erfüllt. Daher können wir durch sukzessive Nachbar-Bildung das gesamte Geschlecht aufzählen. Als Heuristik zur Auswahl, für welchen Knoten als nächstes die Nachbarn konstruiert werden, verwenden wir die Häufigkeit mit der ein Gitter bisher gefunden wurde; unter den am seltensten gefundenen Gittern wird zufällig eines ausgewählt. Es sind noch andere Strategien denkbar, wie beispielsweise eine einfache Breiten- oder Tiefensuche. Des

Weiteren können wir als Abbruchsbedingung das sogenannte *Maß* eines Geschlechtes benutzen (vgl. [Kne02, Abschnitt 35]).

(4.4.3) Definition

Es sei $\mathcal{G}$ ein Geschlecht von Gittern und $L_1, \ldots, L_h$ ein Vertretersystem der Isometrieklassen von Gittern in $\mathcal{G}$. Der Wert

$$\text{Maß}(\mathcal{G}) = \sum_{i=1}^{h} \frac{1}{|\text{Aut}(L_i)|}$$

heißt das *Maß* des Geschlechtes $\mathcal{G}$.

Das Maß eines Geschlechtes kann ohne tatsächliche Aufzählung mithilfe der *Maßformel* berechnet werden. Indem wir die Kehrwerte der Ordnungen der Automorphismengruppen aller bisher gefundenen Gitter während des Algorithmus aufsummieren, können wir also über die Differenz zum tatsächlichen Maß feststellen, ob wir bereits alle Isometrieklassen gefunden haben. Zusätzlich können wir eine weitere nützliche Einschränkung machen: Haben wir bisher Gitter $L_1, \ldots, L_{h'}$ gefunden und ist $m := \sum_{i=1}^{h'} \frac{1}{|\text{Aut}(L_i)|}$, so muss jedes weitere Gitter M im Geschlecht eine Automorphismengruppe mit $|\text{Aut}(M)| \geq \frac{1}{\text{Maß}(\mathcal{G})-m}$ besitzen. Sobald das verbleibende Maß also kleiner als 1 ist, können wir Nachbarn mit zu kleiner Automorphismengruppe übergehen, da diese notwendigerweise isometrisch zu einem bereits gefundenen Gitter sein müssen. Insgesamt kommen wir so auf Algorithmus (6) zur Aufzählung eines Geschlechtes anhand eines Vertreters. Für die Bestimmung eines Vertreters zu einem gegebenen Geschlechtssymbol wird ein `MAGMA`-Programm aus der Diplomarbiet von David Lorch [Lor11] verwendet.

Algorithmus 6 Aufzählung aller Isometrieklassen eines Geschlechtes

Eingabe: Gitter L von Dimension ≥ 3
Ausgabe: Liste von Vertretern aller Isometrieklassen im Geschlecht von L

$\mathcal{G} \leftarrow$ Geschlecht von L
$M \leftarrow \text{Maß}(\mathcal{G})$
$m \leftarrow \frac{1}{|\text{Aut}(L)|}$
$Gen \leftarrow [L]$
$Explored \leftarrow [false]$
$NumFound \leftarrow [1]$
while $m < M$ **do**
    $RareFound \leftarrow \{i \mid \neg Explored[i] \wedge NumFound[i] \leq NumFound[j]\ \forall j\}$
    $i \leftarrow$ zufälliges Element aus $RareFound$
    $Neigh \leftarrow$ 2-Nachbarn von $Gen[i]$
    $Explored[i] \leftarrow true$
    for $N \in Neigh$ **do**
        $MinAuto \leftarrow \frac{1}{M-m}$
        if $|\text{Aut}(N)| < MinAuto$ **then**
            continue
        if $\exists j : Gen[j] \cong N$ **then**
            $NumFound[j] \leftarrow NumFound[j] + 1$
        else
            $Gen \leftarrow Gen \cup [N]$
            $Explored \leftarrow Explored \cup [false]$
            $NumFound \leftarrow NumFound \cup [1]$
            $m \leftarrow m + \frac{1}{|\text{Aut(N)}|}$
return Gen

Für Dimension $n \leq 2$ sind die Voraussetzungen von Satz (4.4.2) nicht erfüllt, dennoch ist die Aufzählung eines Geschlechtes leicht. Das Geschlechtssymbol legt insbesondere die Determinante d aller Gitter des Geschlechtes fest. Im Falle $n = 1$ muss jedes Gitter im Geschlecht folglich die Gram-Matrix $(d) \in \mathbb{Z}^{1\times 1}$ besitzen. Kommen wir nun zum Fall $n = 2$: Die Hermite-Konstante γ_2 hat auf 4 Stellen abgerundet den Wert 1.1547. Für alle Gitter L des Geschlechtes muss also $\mathrm{Min}(L) \leq 1.1548\sqrt{\mathrm{Det}(L)}$ gelten. Sei $t := \mathrm{Min}(L)$ und (e_1, e_2) eine Basis von L mit $b(e_1, e_1) = t$. Die Gram-Matrix von L bezüglich dieser Basis hat die Gestalt $\begin{pmatrix} t & x \\ x & y \end{pmatrix}$. Ohne Einschränkung gilt $-t < x < t$, sonst ersetze sukzessiv e_2 durch $e_2 + e_1$ für $x < -t$, bzw. durch $e_2 - e_1$ für $t < x$, denn $b(e_1, e_2 \pm e_1) = x \pm t$. Der Wert y ist dann eindeutig bestimmt durch $\mathrm{Det}(L) = ty - x^2$. Diese Überlegungen ergeben Algorithmus (7).

Algorithmus 7 Aufzählung aller Isometrieklassen eines Geschlechtes von Dimension 2

Eingabe: Geschlechtssymbol S mit Dimension 2
Ausgabe: Liste von Vertretern aller Isometrieklassen im zugehörigen Geschlecht

$d \leftarrow$ durch S festgelegte Determinante
$Gen \leftarrow []$
for $t \in \{0, \ldots, \lfloor 1.1548\sqrt{d} \rfloor\}$ **do**
    for $x \in \{-t+1, \ldots, t-1\}$ **do**
        $y := \frac{\mathrm{Det}(L) - x^2}{t}$
        if $y \notin \mathbb{Z}$ **then**
            continue
        $L \leftarrow$ Gitter mit Gram-Matrix $\begin{pmatrix} t & x \\ x & y \end{pmatrix}$
        if L hat Geschlechtssymbol S und $\nexists M \in Gen : L \cong M$ **then**
            $Gen \leftarrow Gen \cup [L]$
return Gen

§ 4.5 Konstruktion von Obergittern

Haben wir einen Kandidaten für das Teilgitter $M := L_1 \perp L_p$ und für den Automorphismus $\sigma = \mathrm{diag}(\sigma_1, \sigma_p)$ gefunden, so gilt es, die σ-invarianten Obergitter von M mit Index p^s zu bestimmen. Hierfür verwenden wir je nach Fall verschiedene Methoden. Zunächst ist auf die Konstruktion von Michael Jürgens in [Jü15, Abschnitt (1.4)] hinzuweisen. Hier wird eine Vorgehensweise beschrieben, um mithilfe der isotropen Teilräume des bilinearen $\mathbb{F}_p$-Vektorraums $(M^{\#,p}/M, \bar{b})$, wobei

$$\bar{b} : M^{\#,p}/M \times M^{\#,p}/M \to \mathbb{F}_p, (x + M, y + M) \mapsto pb(x, y) + p\mathbb{Z}$$

die reduzierte Bilinearform auf den Faktorgruppen darstellt, sämtliche ganzen Obergitter von Index p^s zu bestimmen - unabhängig von σ. Da diese Methode wegen Nichtbeachtung der σ-invarianz potentiell noch mehr Gitter liefert als solche mit den geforderten Eigenschaften, ist sie unsere erste Wahl. In einigen Fällen scheitert sie jedoch an der steigenden Ressourcenintensivität.

Nebe verwendet im Beweis von [Neb13, Theorem (5.9)] eine Vorgehensweise, welche die Struktur von M als orthogonale Summe benutzt, um die tatsächlich σ-Invarianten Obergitter zu bestimmen. Diese Vorgehensweise passen wir nun auf unsere Situation an. Dazu bemerken wir zunächst die Tatsache, dass jedes ganze Obergitter $L \geq M$, sodass $[L : M]$ eine p-Potenz ist, ein Teilgitter von $M^{\#,p} = L_1^{\#,p} \perp L_p^{\#,p}$ sein muss. Definiere nun $b_1 := b|_{V_1 \times V_1}$ und $b_p := b|_{V_p \times V_p}$ sowie $\overline{b_1} := \bar{b}|_{(L_1^{\#,p}/L_1) \times (L_1^{\#,p}/L_1)}$ und $\overline{b_p} := \bar{b}|_{(L_p^{\#,p}/L_p) \times (L_p^{\#,p}/L_p)}$. Damit L ein ganzes Gitter wird, muss für beliebige Elemente $(x_1, x_p), (y_1, y_p) \in L$ gelten:

$$\begin{aligned}
0 + p\mathbb{Z} &\stackrel{!}{=} \bar{b}((x_1, x_p) + M, (y_1, y_p) + M) \\
&= pb_1(x_1, y_1) + pb(x_1, y_p) + pb(x_p, y_1) + pb_p(x_p, y_p) + p\mathbb{Z} \\
&= \overline{b_1}(x_1 + M, y_1 + M) + \overline{b_p}(x_p + M, y_p + M).
\end{aligned}$$

Also $\overline{b_1}(x_1+M, y_1+M) = -\overline{b_p}(x_p+M, y_p+M)$. Demnach sind die ganzen Obergitter von Index p^s gegeben durch

$$L_\varphi := \{(x_1, x_p) \in L_1^{\#,p} \perp L_p^{\#,p} \mid \varphi(x_1+L_1) = x_p + L_p\}$$

für die Isometrien $\varphi : (L_1^{\#,p}/L_1, \overline{b_1}) \to (L_p^{\#,p}/L_p, -\overline{b_p})$.

Nun zur σ-Invarianz eines solchen Gitters L_φ. Auf dem Dualgitter $M^\#$ ist σ ebenfalls ein Automorphismus, denn

$$x \in M^\# \Leftrightarrow b(x, M) \subseteq \mathbb{Z} \Leftrightarrow b(\sigma(x), \sigma(M)) \subseteq \mathbb{Z} \Leftrightarrow b(\sigma(x), M) \subseteq \mathbb{Z} \Leftrightarrow \sigma(x) \in M^\#,$$

also gilt auch $\sigma(M^{\#,p}) = M^{\#,p}$ und σ operiert auf $M^{\#,p}/M$ durch $\sigma(x+M) := \sigma(x) + M$. Analog operieren σ_1 auf $L_1^{\#,p}/L_1$ und σ_p auf $L_p^{\#,p}/L_p$. Damit ein Gitter L_φ nun σ-invariant ist, muss für alle $(x_1, x_p) \in L_\varphi$ auch $(\sigma_1(x_1), \sigma_p(x_p)) \in L_\varphi$ sein, also $\varphi(\sigma_1(x_1) + L_1) = \sigma_p(x_p) + L_p = \sigma_p(\varphi(x_1 + L_1))$. Dies führt zur Bedingung

$$\varphi \circ \sigma_1 = \sigma_p \circ \varphi. \tag{4.3}$$

Die Menge aller Isometrien bilden in höheren Dimensionen einen zu großen Suchraum. Wir können allerdings einige Einschränkungen machen. Seien φ_0 und $\varphi_1 : (L_1^{\#,p}/L_1, \overline{b_1}) \to (L_p^{\#,p}/L_p, -\overline{b_p})$ Isometrien, welche Bedingung (4.3) erfüllen. Dann gilt

$$\sigma_1 \varphi_0 \varphi_1^{-1} \sigma_1^{-1} = (\sigma_1 \varphi_0)(\sigma_1 \varphi_1)^{-1} = \varphi_0 \sigma_p \sigma_p^{-1} \varphi_1^{-1} = \varphi_0 \varphi_1^{-1}.$$

Demnach ist $\varphi_0 \varphi_1^{-1}$ enthalten im Zentralisator $C_{O(L_1^{\#,p}/L_1)}(\sigma_1)$. Fixieren wir also eine Isometrie φ_0, so erhalten wir alle weiteren durch Verkettung mit Zentralisatorelementen. `MAGMA` kann eine einzelne Isometrie $\tilde{\varphi_0}$ konstruieren, diese erfüllt jedoch nicht notwendigerweise Bedingung (4.3). Wir machen den Ansatz $\varphi_0 \stackrel{!}{=} \tilde{\varphi_0} \circ u$ für $u \in O(L_1^{\#,p}/L_1)$. Dann muss gelten

$$\begin{aligned} & \varphi_0 \circ \sigma_1 = \sigma_p \circ \varphi_0 \\ \Leftrightarrow \quad & \tilde{\varphi_0} \circ u \circ \sigma_1 = \sigma_p \circ \tilde{\varphi_0} \circ u \end{aligned}$$

$$\Leftrightarrow \quad u \circ \sigma_1 \circ u^{-1} = \tilde{\varphi_0} \circ \sigma_p \circ \tilde{\varphi_0}^{-1}.$$

Das Element u konjugiert also die Abbildung σ_1 zu $\tilde{\varphi_0} \circ \sigma_p \circ \tilde{\varphi_0}^{-1}$. Ein solches Element lässt sich durch `MAGMA` bestimmen.

Eine letzte Beobachtung ist die Folgende: Sei φ eine Isometrie mit den gesuchten Eigenschaften und $c \in C_{\mathrm{Aut}(L_1)}(\sigma_1)$, dann operiert c auf $L_1^{\#,p}/L_1$ und die Abbildung $\varphi_c := \varphi \circ c$ ist ebenfalls eine Isometrie mit Eigenschaft (4.3), da c im Zentralisator von σ_1 liegt. Diese Abbildung liefert somit ebenfalls ein σ-invariantes ganzes Obergitter L_{φ_c} von M. Definiere nun die Isometrie

$$\psi : L_1^{\#,p} \perp L_p^{\#,p} \to L_1^{\#,p} \perp L_p^{\#,p}, (x, y) \mapsto (c(x), y),$$

dann ist $\psi(L_{\varphi_c}) = L_\varphi$, also sind die Gitter isometrisch. Um Redundanz bei der Konstruktion zu vermeiden, genügt es daher, jeweils einen Vertreter nach den Restklassen modulo $C_{\mathrm{Aut}(L_1)}(\sigma_1)$ zu wählen.

Insgesamt erhalten wir also Algorithmus (8):

Algorithmus 8 Konstruktion von σ-invarianten Obergittern

Eingabe: Gitter L_1, L_p, Automorphismen σ_1 von L_1 und σ_p von L_p, $p \in \mathbb{P}$, sodass $L_1^{\#,p}/L_1 \cong (\mathbb{F}_p)^s \cong L_p^{\#,p}/L_p$

Ausgabe: Liste aller nicht-isometrischen, ganzen Obergitter von $L_1 \perp L_p$ von Index p^s

$\tilde{\varphi_0} \leftarrow$ Isometrie $(L_1^{\#,p}/L_1, \overline{b_1}) \to (L_p^{\#,p}/L_p, -\overline{b_p})$

$u \leftarrow$ Element in $O(L_1^{\#,p}/L_1)$, sodass u $\circ \sigma_1 \circ u^{-1} = \tilde{\varphi_0} \circ \sigma_p \circ \tilde{\varphi_0}^{-1}$

$\varphi_0 \leftarrow \tilde{\varphi_0} \circ u$

$C \leftarrow$ Vertretersystem von $C_{O(L_1^{\#,p}/L_1)}(\sigma_1)/C_{\mathrm{Aut}(L_1)}(\sigma_1)$

$\mathit{Results} \leftarrow []$

for $c \in C$ **do**

  $\varphi \leftarrow \varphi_0 \circ c$

  $L_\varphi \leftarrow \{(x_1, x_p) \in L_1^{\#,p} \perp L_p^{\#,p} \mid \varphi(x_1 + L_1) = x_p + L_p\}$

  if $\nexists M \in \mathit{Results} \mid L_\varphi \cong M$ **then**

    $\mathit{Results} \leftarrow \mathit{Results} \cup [L_\varphi]$

return $\mathit{Results}$

Es ist anzumerken, dass auch diese Methode in zu hohen Dimensionen zu ineffizient wird. Als letzte Möglichkeit können wir deshalb mit der `MAGMA`-Methode `Sublattices` σ-invariante Teilgitter von $M^{\#,p}$ bestimmen. Da die konstruierten Gitter in diesem Fall jedoch nicht notwendigerweise ganz sein müssen, steigt die Anzahl schnell an und es kann in der Regel keine vollständige Liste gebildet werden, sondern nur eine Teilmenge der Gitter. Bei Benutzung dieser Methode kann also die Vollständigkeit der Ergebnisse nicht garantiert werden.

§ 4.6 Konstruktion von Gittern mit großem Automorphismus

Mithilfe der Typen von Automorphismen mit Primzahlordnung kennen wir die Geschlechter - oder zumindest Dimension und Determinante - von Fix- und Bild-Gitter. In der Regel ist mindestens eines der Geschlechter zu groß, um es mithilfe der Kneser-Methode in akzeptabler Zeit aufzuzählen. Hat das L jedoch einen großen Automorphismus, so kann L_p eine Ideal-Gitter-Gestalt besitzen, wodurch die Konstruktion erleichtert wird. Wir untersuchen zunächst, welche Form die Potenzen der Minimalpolynome von Automorphismen besitzen.

(4.6.1) Lemma

Ist V ein Vektorraum und $\sigma \in GL(V)$ mit Minimalpolynom $\mu_\sigma = \Phi_{n_1}\Phi_{n_2}\dots\Phi_{n_k}$, dann hat σ^d für $d \in \mathbb{N}$ das Minimalpolynom

$$\mu_{\sigma^d} = \mathrm{kgV}(\Phi_{n_1/\mathrm{ggT}(n_1,d)}, \dots, \Phi_{n_k/\mathrm{ggT}(n_k,d)})$$

Beweis:

Sei zunächst $k = 1$, also $\mu_\sigma = \Phi_{n_1}$. Dann ist $|\langle\sigma^d\rangle| = \frac{n_1}{\mathrm{ggT}(n_1,d)}$, also ist σ^d eine primitive Einheitswurzel mit $\mu_{\sigma^d} = \Phi_{n_1/\mathrm{ggT}(n_1,d)}$. Für $k > 1$ können wir V zerlegen zu

$$V = \mathrm{Kern}(\Phi_{n_1}(\sigma)) \oplus \mathrm{Kern}(\Phi_{n_2}(\sigma)) \oplus \dots \oplus \mathrm{Kern}(\Phi_{n_k}(\sigma)) =: V_1 \oplus \dots \oplus V_k$$

Somit hat $\sigma|_{V_i}$ jeweils das Minimalpolynom Φ_{n_i} für $i = 1, \dots, k$. Es folgt:

$$\mu_{\sigma^d} = \mathrm{kgV}(\mu_{\sigma^d|_{V_1}}, \dots, \mu_{\sigma^d|_{V_k}}) = \mathrm{kgV}(\Phi_{n_1/\mathrm{ggT}(n_1,d)}, \dots, \Phi_{n_k/\mathrm{ggT}(n_k,d)}). \qquad \square$$

Sei nun L ein Gitter der Dimension n und $\sigma \in \mathrm{Aut}(L)$ ein großer Automorphismus von Ordnung m. Dann hat das Minimalpolynom von σ die Form $\mu_\sigma = \Phi_m\Phi_{n_1}\dots\Phi_{n_k}$.

Ist $\mathrm{kgV}(n_1,\ldots,n_k) < m$, so existiert eine Primzahl p mit $\mathrm{kgV}(n_1,\ldots,n_k)|\frac{m}{p} =: d$. Der Automorphismus σ^d hat Primzahlordnung p und liefert wie in Abschnitt (4.2) eine σ-invariante Zerlegung

$$V = \mathrm{Bild}(\sigma^d - 1) \oplus \mathrm{Kern}(\sigma^d - 1).$$

Nun ist allerdings $\mathrm{Kern}(\sigma^d - 1) = \mathrm{Kern}\left((\Phi_{n_1} \ldots \Phi_{n_k})(\sigma)\right)$ und somit $\mathrm{Bild}(\sigma^d - 1) = \mathrm{Kern}(\Phi_m(\sigma))$. Das zu σ^d gehörige Bildgitter $L_p = L \cap \mathrm{Bild}(\sigma^d - 1) = L \cap \mathrm{Kern}(\Phi_m(\sigma))$ ist also ein Sub-Ideal-Gitter von L. Das Fix-Gitter $L_1 = L \cap \mathrm{Kern}(\sigma^d - 1)$ hat die Dimension $n - \varphi(m) < \frac{n}{2}$.

Wir betrachten nun das Minimalpolynom der Operation von σ auf den Faktorgruppen $L_1^{\#,p}/L_1$ und $L_p^{\#,p}/L_p$. Nach (4.2.4) waren $L_1^{\#,p}/L_1$ und $L_p^{\#,p}/L_p$ isomorph und der zugehörige Isomorphismus war die Komposition der drei Isomorphismen

$$\begin{aligned} L_1^{\#,p}/L_1 &\to L_1^{\#}/L_1^{\#,\ell}, x + L_1 \mapsto x + L_1^{\#,\ell} \\ L_1^{\#}/L_1^{\#,\ell} &\to L_p^{\#}/L_p^{\#,\ell}, y + L_1^{\#,\ell} \mapsto (\hat{y} - y) + L_p^{\#,\ell} \\ L_p^{\#}/L_p^{\#,\ell} &\to L_p^{\#,p}/L_p, x + L_p^{\#,\ell} \mapsto x + L_p, \end{aligned}$$

wobei $\hat{y} \in L^{\#}$ mit $b(x,y) = b(x,\hat{y})$ für alle $x \in L_1$. Man sieht leicht ein, dass $\sigma(\hat{y})$ eine mögliche Wahl für $\widehat{\sigma(y)}$ darstellt, da alle beteiligten Gitter und die Bilinearform invariant unter σ sind. Alle drei Isomorphismen kommutieren also mit σ. Daraus folgt, dass die Faktorgruppen auch als $\mathbb{Z}[\sigma]$-Moduln, bzw. wegen $pL_p^{\#,p} \subseteq L_p$ und $pL_1^{\#,p} \subseteq L_1$ sogar als $\mathbb{F}_p[\sigma]$-Moduln isomorph sind. Insbesondere hat σ auf $L_1^{\#,p}/L_1$ und $L_p^{\#,p}/L_p$ dasselbe Minimalpolynom. Wegen $(1-\sigma^d)L_p^{\#,p} \subseteq L_p$ wird $L_p^{\#,p}/L_p$ zu einem $\mathbb{F}_p[\sigma]/(1-\sigma^d) \cong \mathbb{F}_p[\zeta_d]$-Modul. Das Minimalpolynom der Operationen von σ auf den Faktorgruppen ist somit ein Teiler von Φ_d, also (falls $\mathrm{Dim}_{\mathbb{F}_p}(L_p^{\#,p}/L_p) = \mathrm{Dim}(L_1^{\#,p}/L_1) > 0$) gleich Φ_d. In diesem Falle muss daher

$$\mathrm{Grad}(\mu_{\sigma|_{\mathrm{Bild}(\sigma^d-1)}}), \mathrm{Grad}(\mu_{\sigma|_{\mathrm{Kern}(\sigma^d-1)}}) \geq \mathrm{Grad}(\Phi_d) = \varphi(d)$$

erfüllt sein. Außerdem gilt $\varphi(d) \leq \mathrm{Dim}_{\mathbb{F}_p}(L_p^{\#,p}/L_p) = s$.

Wir entwerfen nun den Algorithmus (9) zur Konstruktion solcher Gitter L.

Algorithmus 9 Konstruktion von Gittern mit großem Automorphismus

Eingabe: $n \in \mathbb{N}$, quadratfreies $\ell \in \mathbb{N}$, $m \in \mathbb{N}$ mit $\frac{n}{2} < \varphi(m) \leq n$.

Ausgabe: Liste von extremalen ℓ-modularen Gittern der Dimension n mit einem großen Automorphismus σ der Ordnung m, sodass ein $p \in \mathbb{P}$, $\mathrm{ggT}(p, \ell) = 1$ existiert mit $\frac{\mu_\sigma}{\Phi_m} | (X^{\frac{m}{p}} - 1)$

$Results \leftarrow []$

$AutoTypes \leftarrow$ Liste von Aut.-Typen von Primzahlordnung nach Algorithmus (5)

for $p \in \{q \in \mathbb{P} \mid q | m, \mathrm{ggT}(q, \ell) > 1\}$ **do**

  $d \leftarrow \frac{m}{p}$

  $PossibleTypes \leftarrow \{p - (n - \varphi(m), \varphi(m)) - s - \cdots \in AutoTypes \mid s = 0$ oder $\varphi(d) \leq s\}$

  for $t \in PossibleTypes$ **do**

    $L_p\_List \leftarrow$ Liste von Ideal-Gittern über $\mathbb{Q}(\zeta_m)$ mit durch t für L_p vorgegebene Dimension und Determinante nach Algorithmus (4)

    if $s = 0$ **then return** L_p_List

    for $L_p \in L_p\_List$ **do**

      $L_1\_List \leftarrow$ Liste von allen Gittern mit durch t für L_1 vorgebenene Dimension und Determinante und mit quadratfreier Stufe nach Algorithmus (6)

      for $L_1 \in L_1\_List$ **do**

        for $\sigma_p \in \{\sigma \in \mathrm{Aut}(L_p) \mid \sigma$ op. auf $L_p^{\#,p}/L_p$ mit Mi.-Po. $\mathbf{\Phi}_d\}$ **do**

          for $\sigma_1 \in \{\sigma \in \mathrm{Aut}(L_1) \mid \sigma$ op. auf $L_1^{\#,p}/L_1$ mit Mi.-Po. $\mathbf{\Phi}_d$ und $\mathrm{Grad}(\mu_\sigma) \leq \varphi(d)\}$ **do**

            $M \leftarrow L_1 \perp L_p$

            $\sigma \leftarrow \mathrm{diag}(\sigma_1, \sigma_p)$

            $L\_List \leftarrow$ Liste σ-invarianter Obergitter von M mit Index p^s

            for $L \in L\_List$ **do**

              if L ist ℓ-modulares extremales Gitter **then**

                $Results \leftarrow Results \cup [L]$

return $Results$

Für $p = 2$ kann bei Zeile 12 des Algorithmus alternativ auch mit Lemma (4.2.6) vorgegangen werden: Man zählt stattdessen die möglichen U auf und erhält die Kandidaten für L_1 als die Obergitter von $\sqrt{2}U$ vom Index $p^{\frac{\varphi(n)-m-s}{2}}$ mit quadratfreier Stufe und ausreichend großem Minimum.

Für diesen Algorithmus wurden Einschränkungen an die Minimalpolynome der Automorphismen von Gittern gemacht, es ist also a priori nicht klar, welcher Grad von Vollständigkeit durch die Klassifikation mit unserem Algorithmus gegeben ist. Dennoch ist es gelungen, einige neue extremale Gitter zu konstruieren. Die Anzahlen gefundener Gitter sind in Tabelle (4.6) festgehalten. Für $(\ell, n) \in \{(7, 18), (7, 20), (1, 24), (2, 24), (5, 24)\}$ wurden die Obergitter σ-invariant konstruiert und nicht allgemein mit Jürgens Methode. Bei $\ell = 7$, $n = 20$ konnte in einem Falle nicht das gesamte Geschlecht für L_1 aufgezählt werden. Zur tatsächlichen Masse fehlte $\frac{167}{4644864}$. Das gleiche Problem trat bei $\ell = 7$, $n = 22$ auf. Hier fehlte bei einem Geschlecht $\frac{167}{4644864}$.

$n \backslash \ell$	1	2	3	5	6	7	11	14	15
4	–	–	1	–	–	–	1	–	–
6	–	–	1	–	–	1	2	–	–
8	1	1	1	1	–	1	1	1	1
10	–	–	1	–	–	–	–	–	–
12	–	1	1	1	1	–	–	1	1
14	–	–	1	–	–	–	–	–	–
16	2	1	2	–	1	3	–	–	1
18	–	–	1	–	–	–	–	–	–
20	–	1	3	–	1	–	–	–	–
22	–	–	$3(1^*)$	–	–	–	–	–	–
24	1	$5(2^*)$	1	1	$5(3^*)$	–	–	–	–
26	–	–	1	–	–	–	–	–	–
28	–	$27(25^*)$	$3(2^*)$	–	–	–	–	–	–

Tabelle 4.1: Anzahl der durch Algorithmus (9) konstruierten extremalen stark ℓ-modularen Gitter in Dimension $n \leq 28$ sowie ggf. der Anzahl der bisher unbekannten Gitter darunter

§ 4.7 Vollständigkeit der Ergebnisse

In diesem letzten Abschnitt wollen wir die Gitter genauer untersuchen, die von Algorithmus (9) nicht gefunden werden. Genauer: Wir klassifizieren die Möglichkeiten für die charakteristischen Polynome von Gittern, die durch den Algorithmus nicht konstruiert werden können. Dazu untersuchen wir für die Menge aller Kandidaten für charakteristische Polynome die "Kompatiblität" der Dimensionen der von σ-Potenzen induzierten

Fix- und Bildgitter.

Sei σ eine Isometrie eines n-dimensionalen Vektorraums V mit Ordnung $|\sigma| = m$, dann ist

$$\chi_\sigma = \Phi_{d_1}^{c_1} \dots \Phi_{d_k}^{c_k}$$

für die Teiler d_i von m und gewisse Potenzen c_i. Die Dimension eines Hauptraumes $\mathrm{Kern}(\Phi_{d_i}(\sigma))$ ist in diesem Falle $c_i\varphi(d_i)$ und ähnlich wie in Lemma (4.6.1) ist

$$\mathrm{Dim}(\mathrm{Kern}(\sigma^d - 1)) = \sum_{d_i | d} c_i\varphi(d_i)$$

für alle $d \in \mathbb{N}_0$ und $n = \sum_{i=1}^k c_i\varphi(d_i)$. Kennt man daher für Primteiler $p_1, \dots, p_t \mid m$ die Typen

$$\begin{gathered} p_1 - (n_{1,1}, \dots \\ p_2 - (n_{2,1}, \dots \\ \vdots \\ p_t - (n_{t,1}, \dots \end{gathered}$$

der Automorphismen $\sigma^{\frac{m}{p_1}}, \sigma^{\frac{m}{p_2}}, \dots, \sigma^{\frac{m}{p_t}}$, so muss der Vektor $c := (c_1, \dots, c_k) \in \mathbb{N}_0^k$ eine Lösung des Gleichungssystems $cM = (n_{1,1}, n_{2,1}, \dots, n_{t,1}, n)$ mit der Matrix

$$M \in \mathbb{N}_0^{k\times(t+1)}, \quad M_{i,j} := \begin{cases} \varphi(d_i) & , d_i \mid \frac{m}{p_j} \text{ oder } j = t+1 \\ 0 & , \text{sonst} \end{cases}$$

sein. Außerdem gilt $\mathrm{kgV}\{d_i | c_i > 0\} = m$.

Von den verbleibenden Kandidaten für das charakteristische Polynom werden nun alle diejenigen potentiell *nicht* von Algorithmus (9) gefunden, für die gilt, dass

$$\mathrm{kgV}\{d_i \mid d_i \neq m \text{ und } c_i > 0\} = m.$$

Diese Charakterisierung der möglichen charakteristischen Polynome allein anhand einer Menge von Typen wenden wir nun einmal beispielhaft auf 3-modulare Gitter der Dimension 24 an.

(4.7.1) Satz
Sei L ein extremales 3-modulares Gitter in einem bilinearen Vektorraum (V, b) der Dimension 24. Dann hat L keine Automorphismen der Ordnung 7 sowie der Ordnung $p \in P_{\geq 13}$. Ist $\sigma \in \mathrm{Aut}(L)$ von Ordnung m mit $12 < \varphi(m) \leq 24$, so ist $m \in \{60, 27, 33, 66, 45, 72, 90\}$. Außerdem gelten folgende Einschränkungen:

- $m = 60 \Rightarrow \Phi_{20} | \mu_\sigma$.
- $m = 27 \Rightarrow \Phi_{27} | \mu_\sigma$.
- $m = 33 \Rightarrow \Phi_3 \Phi_{11} | \mu_\sigma$.
- $m = 66 \Rightarrow \Phi_{11} \Phi_{22} | \mu_\sigma$.
- $m = 45 \Rightarrow \Phi_5 \Phi_9 | \mu_\sigma$.
- $m = 72 \Rightarrow \Phi_8 | \mu_\sigma$.
- $m = 90 \Rightarrow \Phi_{10} \Phi_{15} | \mu_\sigma$.

Beweis:
Es muss $\mathrm{Min}(L) \geq 6$ sein. Algorithmus (5) liefert uns 50 mögliche Automorphismentypen. Zählt man einige der Geschlechter mit den durch die Typen festgelegten Dimensionen und Determinanten auf, so sieht man, dass in vielen Fällen darin keine Gitter mit Minimum ≥ 6 enthalten sind, wodurch diese Typen ausgeschlossen werden können. Die verbleibenden Typen von Automorphismen der Ordnung $p \in \mathbb{P} - \{3\}$ sind:

$2-(12,12)-6-(6,6)$	$2-(12,12)-8-(8,4)$
$2-(12,12)-8-(4,8)$	$2-(12,12)-10-(8,4)$
$2-(12,12)-10-(4,8)$	$2-(12,12)-12-(10,2)$
$2-(12,12)-12-(8,4)$	$2-(12,12)-12-(6,6)$

$$
\begin{aligned}
&2-(12,12)-12-(4,8) && 2-(12,12)-12-(2,10)\\
&2-(0,24)-0-(0,12) && 5-(8,16)-4-(8,4)\\
&5-(8,16)-4-(4,8) && 5-(0,24)-0-(0,12)\\
&7-(0,24)-0-(0,12) && 11-(4,20)-2-(2,10)\\
&13-(0,24)-0-(0,12).
\end{aligned}
$$

Insbesondere stellt man fest, dass keine Automorphismen der Ordnung > 13 existieren. Mit dieser Einschränkung zusammen mit der Bedingung $12 < \varphi(m) \leq 24$ muss m in der Menge

$$\{32, 40, 48, 60, 27, 54, 25, 33, 44, 50, 66, 45, 72, 90\}$$

liegen. Für diese Fälle zählen wir mithilfe der Automorphismentypen mit dem in diesem Abschnitt beschriebenen Verfahren die möglichen charakteristischen Polynome auf. Eine genaue Betrachtung der Möglichkeiten für die Polynome zeigt alle verbleibenden Aussagen. $\square$

5 Zusammenfassung und Ausblick

Nach einem einführenden Kapitel zum Begriff eines modularen Gitters und zur Dichte eines Gitters haben wir die Theorie der Ideal-Gitter dargelegt, welche in einem Algorithmus von Jürgens in [Jü15] bereits klassifiziert werden konnten. Wir haben die Schritte des Algorithmus von Jürgens genau beschrieben und erfolgreich in `MAGMA` implementiert. So konnten die Ergebnisse von Jürgens rekonstruiert und auf einige - bisher nicht von Jürgens aufgezählte - Dimensionen erweitert werden.

In Kapitel 4 haben wir zunächst die von Jürgens und Nebe beschriebene Theorie zu den Typen von Gitterautomorphismen mit Primzahlordnung aufgegriffen sowie in gewissen Fällen verallgemeinert. Anschließend wurde der Begriff eines Gittergeschlechtes nach [CS93] eingeführt und das Kneser'sche Nachbarschaftsverfahren erläutert, mit dessen Hilfe wir relevante Geschlechter aufzählen können. Des Weiteren haben wir Methoden von Jürgens und Nebe auf unsere Situation angepasst, mit deren Hilfe wir ganzzahlige (σ-invariante) Obergitter konstruieren können. Alle beschriebenen Aussagen und Methoden führten wir nun zu einem finalen Algorithmus zusammen, der gewisse extremale modulare Gitter mit großen Automorphismen konstruieren kann. In der aktuellen Fassung unterliegt der Algorithmus noch einigen Einschränkungen. Besonders die Tatsache, dass wir im Zuge dieser Arbeit noch nichts über Automorphismen von Primzahlordnung p aussagen konnten, sodass p ein Teiler der Stufe des Gitters ist, stellt eine große Erschwerung dar und verhindert in vielen Fällen eine vollständigere Klassifikation. Im letzten Abschnitt der Arbeit haben wir schließlich die charakteristischen Polynome all solcher Gitter klassifiziert, die durch den bestehenden Algorithmus gegebenenfalls *nicht* gefunden werden können.

Ein Ansatzpunkt für weitere Forschung zu diesem Thema ist insbesondere die Untersuchung von Automorphismen mit nicht zur Stufe teilerfremder Ordnung. Eine Charakterisierung solcher Automorphismen könnte den Algorithmus potentiell deutlich verbessern. Des Weiteren ist eine Optimierung der Teilschritte von Interesse. Besonders die Konstruktion eines Repräsentanten sowie die vollständige Aufzählung von Geschlechtern versagt in hohen Dimensionen schnell an Hardware-Limitierungen. Zumindest in Spezialfällen ist eventuell eine effizientere Berechnung möglich. Zuletzt könnte möglicherweise die Klassifikation der nicht-gefundenen Gitter allgemein oder in Spezialfällen verfeinert werden.

6 Anhang

§ 6.1 Ergebnisse der Ideal-Gitter-Klassifikation

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
1	8	1(1)	$\mathbb{Q}(\zeta_{15})$	1	–	–	–	–	–	–	–	–
	16	1(1)	$\mathbb{Q}(\zeta_{40})$	1	–	–	–	–	–	–	–	–
	24	4(1)	$\mathbb{Q}(\zeta_{35})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{45})$	1	–	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{54})$	1	–	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{75})$	1	–	–	–	–	–	–	–	–
	32	7(5)	$\mathbb{Q}(\zeta_{51})$	–	2	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{68})$	1	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{80})$	1	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	1	–	–	–	–	–	–	–
2	4	1(1)	$\mathbb{Q}(\zeta_{8})$	1	–	–	–	–	–	–	–	–
	8	1(1)	$\mathbb{Q}(\zeta_{16})$	1	–	–	–	–	–	–	–	–
	12	1(1)	$\mathbb{Q}(\zeta_{36})$	1	–	–	–	–	–	–	–	–

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
2	16	2(1)	$\mathbb{Q}(\zeta_{32})$	1	–	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{40})$	–	1	–	–	–	–	–	–	–
	20	1(1)	$\mathbb{Q}(\zeta_{33})$	–	1	–	–	–	–	–	–	–
	24	2(1)	$\mathbb{Q}(\zeta_{56})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	1	–	–	–	–	–	–	–	–
	32	13(4)	$\mathbb{Q}(\zeta_{51})$	–	–	3	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{64})$	1	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{68})$	–	3	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{80})$	–	1	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{96})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	2	–	–	–	–	–	–	–
	36	6(3)	$\mathbb{Q}(\zeta_{57})$	–	–	3	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{76})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{108})$	1	1	–	–	–	–	–	–	–
3	4	1(1)	$\mathbb{Q}(\zeta_{12})$	1	–	–	–	–	–	–	–	–
	6	1(1)	$\mathbb{Q}(\zeta_{9})$	1	–	–	–	–	–	–	–	–
	8	1(1)	$\mathbb{Q}(\zeta_{24})$	1	–	–	–	–	–	–	–	–
	12	2(1)	$\mathbb{Q}(\zeta_{21})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{36})$	1	–	–	–	–	–	–	–	–
	16	3(2)	$\mathbb{Q}(\zeta_{40})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	1	–	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{60})$	–	1	–	–	–	–	–	–	–
	18	1(–)	$\mathbb{Q}(\zeta_{27})$	1	–	–	–	–	–	–	–	–
	24	7(1)	$\mathbb{Q}(\zeta_{39})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{52})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{56})$	–	2	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	1	1	–	–	–	–	–	–	–

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
3	32	13(7)	$\mathbb{Q}(\zeta_{80})$	–	2	4	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{96})$	1	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	3	2	–	–	–	–	–	–
	36	8(–)	$\mathbb{Q}(\zeta_{57})$	–	1	2	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{63})$	–	1	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{76})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{108})$	1	–	1	–	–	–	–	–	–
5	8	1(1)	$\mathbb{Q}(\zeta_{15})$	–	1	–	–	–	–	–	–	–
	12	1(1)	$\mathbb{Q}(\zeta_{21})$	–	1	–	–	–	–	–	–	–
	16	1(–)	$\mathbb{Q}(\zeta_{40})$	–	1	–	–	–	–	–	–	–
	24	5(1)	$\mathbb{Q}(\zeta_{35})$	–	–	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{45})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{56})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{84})$	–	1	–	–	–	–	–	–	–
	32	10(–)	$\mathbb{Q}(\zeta_{80})$	–	1	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{96})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	–	2	5	–	–	–	–	–
	36	8(–)	$\mathbb{Q}(\zeta_{63})$	–	1	–	7	–	–	–	–	–
6	8	1(1)	$\mathbb{Q}(\zeta_{24})$	–	1	–	–	–	–	–	–	–
	12	1(1)	$\mathbb{Q}(\zeta_{28})$	–	1	–	–	–	–	–	–	–
	16	2(1)	$\mathbb{Q}(\zeta_{40})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	–	1	–	–	–	–	–	–	–
	20	1(1)	$\mathbb{Q}(\zeta_{33})$	–	–	1	–	–	–	–	–	–
	24	5(2)	$\mathbb{Q}(\zeta_{56})$	–	1	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	–	1	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{84})$	–	–	–	1	–	–	–	–	–

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
6	32	12(–)	$\mathbb{Q}(\zeta_{80})$	–	–	1	5	–	–	–	–	–
			$\mathbb{Q}(\zeta_{96})$	–	1	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	–	–	4	–	–	–	–	–
7	6	1(1)	$\mathbb{Q}(\zeta_{7})$	–	1	–	–	–	–	–	–	–
	8	1(1)	$\mathbb{Q}(\zeta_{24})$	–	1	–	–	–	–	–	–	–
	12	1(–)	$\mathbb{Q}(\zeta_{28})$	–	1	–	–	–	–	–	–	–
	16	4(3)	$\mathbb{Q}(\zeta_{40})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	–	1	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{60})$	–	–	1	–	–	–	–	–	–
	20	1(–)	$\mathbb{Q}(\zeta_{44})$	–	–	1	–	–	–	–	–	–
	24	8(–)	$\mathbb{Q}(\zeta_{56})$	–	1	2	2	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	–	1	1	–	–	–	–	–	–
	32	19(–)	$\mathbb{Q}(\zeta_{80})$	–	–	1	1	2	–	–	–	–
			$\mathbb{Q}(\zeta_{96})$	–	1	2	3	–	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	–	2	7	–	–	–	–	–
11	4	1(1)	$\mathbb{Q}(\zeta_{12})$	–	1	–	–	–	–	–	–	–
	8	2(1)	$\mathbb{Q}(\zeta_{15})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{24})$	–	1	–	–	–	–	–	–	–
	10	1(1)	$\mathbb{Q}(\zeta_{11})$	–	–	1	–	–	–	–	–	–
	12	1(–)	$\mathbb{Q}(\zeta_{36})$	–	1	–	–	–	–	–	–	–
	16	5(–)	$\mathbb{Q}(\zeta_{40})$	–	–	1	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{60})$	–	–	–	2	–	–	–	–	–
	20	2(–)	$\mathbb{Q}(\zeta_{33})$	–	–	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{44})$	–	–	1	–	–	–	–	–	–

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
11	24	7(−)	$\mathbb{Q}(\zeta_{35})$	–	–	–	1	–	1	–	–	–
			$\mathbb{Q}(\zeta_{45})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{56})$	–	–	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	–	1	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{84})$	–	–	1	–	–	–	–	–	–
	32	42(−)	$\mathbb{Q}(\zeta_{80})$	–	–	1	1	1	1	–	–	–
			$\mathbb{Q}(\zeta_{96})$	–	1	–	–	1	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	–	1	13	18	4	–	–	–
	36	2(−)	$\mathbb{Q}(\zeta_{108})$	–	1	–	–	1	–	–	–	–
14	4	1(1)	$\mathbb{Q}(\zeta_{8})$	–	1	–	–	–	–	–	–	–
	8	2(1)	$\mathbb{Q}(\zeta_{16})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{24})$	–	–	1	–	–	–	–	–	–
	12	1(1)	$\mathbb{Q}(\zeta_{28})$	–	–	–	1	–	–	–	–	–
	16	5(−)	$\mathbb{Q}(\zeta_{32})$	–	1	–	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{40})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	–	–	1	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{60})$	–	–	1	–	–	–	–	–	–
	24	8(−)	$\mathbb{Q}(\zeta_{56})$	–	–	–	4	–	2	–	–	–
			$\mathbb{Q}(\zeta_{72})$	–	–	1	1	–	–	–	–	–
	32	21(−)	$\mathbb{Q}(\zeta_{64})$	–	1	–	–	2	–	–	–	–
			$\mathbb{Q}(\zeta_{80})$	–	–	–	–	–	2	1	–	–
			$\mathbb{Q}(\zeta_{96})$	–	–	1	1	2	2	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	–	–	4	–	5	–	–	–
	36	36(−)	$\mathbb{Q}(\zeta_{57})$	–	–	–	–	3	25	8	–	–

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
15	8	1(1)	$\mathbb{Q}(\zeta_{24})$	–	–	1	–	–	–	–	–	–
	16	3(1)	$\mathbb{Q}(\zeta_{40})$	–	–	–	–	1	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	–	–	1	–	–	–	–	–	–
			$\mathbb{Q}(\zeta_{60})$	–	–	–	1	–	–	–	–	–
	24	5(–)	$\mathbb{Q}(\zeta_{56})$	–	–	–	1	–	–	–	–	–
			$\mathbb{Q}(\zeta_{72})$	–	–	1	–	–	1	–	–	–
			$\mathbb{Q}(\zeta_{84})$	–	–	–	–	–	2	–	–	–
	32	23(–)	$\mathbb{Q}(\zeta_{80})$	–	–	–	–	2	3	1	–	–
			$\mathbb{Q}(\zeta_{96})$	–	–	1	–	1	–	–	–	–
			$\mathbb{Q}(\zeta_{120})$	–	–	–	4	1	9	1	–	–
	36	4(–)	$\mathbb{Q}(\zeta_{76})$	–	–	–	–	2	1	–	1	–
23	4	1(1)	$\mathbb{Q}(\zeta_{12})$	–	–	1	–	–	–	–	–	–
	8	3(–)	$\mathbb{Q}(\zeta_{24})$	–	–	1	2	–	–	–	–	–
	12	1(–)	$\mathbb{Q}(\zeta_{36})$	–	–	1	–	–	–	–	–	–
	16	5(–)	$\mathbb{Q}(\zeta_{40})$	–	–	–	–	1	–	–	–	–
			$\mathbb{Q}(\zeta_{48})$	–	–	1	2	–	–	–	–	–
			$\mathbb{Q}(\zeta_{60})$	–	–	–	–	–	1	–	–	–
	22	2(–)	$\mathbb{Q}(\zeta_{23})$	–	–	–	–	–	2	–	–	–

ℓ	Dim	Gesamtzahl(extremal)	K	Minimum								
				2	4	6	8	10	12	14	16	18
23	24	14(–)	$\mathbb{Q}(\zeta_{39})$	–	–	–	–	1	1	–	1	–
			$\mathbb{Q}(\zeta_{52})$	–	–	–	–	1	2	–	–	–
			$\mathbb{Q}(\zeta_{56})$	–	–	–	–	–	–	–	1	–
			$\mathbb{Q}(\zeta_{72})$	–	–	1	2	1	2	–	–	–
			$\mathbb{Q}(\zeta_{84})$	–	–	–	–	–	1	–	–	–
	32	20(–)	$\mathbb{Q}(\zeta_{80})$	–	–	–	–	1	–	1	1	1
			$\mathbb{Q}(\zeta_{96})$	–	–	1	2	–	–	2	2	–
			$\mathbb{Q}(\zeta_{120})$	–	–	–	–	1	3	1	4	–
	36	2(–)	$\mathbb{Q}(\zeta_{108})$	–	–	1	–	–	–	1	–	–

Tabelle 6.1: Anzahlen der Ideal-Gitter der Stufen $\ell \in \{1, 2, 3, 5, 6, 7, 11, 14, 15, 23\}$ und Determinante $\ell^{\frac{n}{2}}$ mit Dimensionen ≤ 36 nach zugehörigem Kreisteilungskörper K und Minimum.

§ 6.2 MAGMA-Implementierungen von Hilfsfunktionen

Es folgt der Quellcode zu folgenden Hilfsfunktionen in der Reihenfolge des Quellcodes:

- Das komplex-konjugierte eines $\mathbb{Z}_K$-Ideals berechnen.
- Eine Liste von Gittern nach Isometrie reduzieren.
- Das Minimum ausgeben, was ein ℓ-modulares Gitter der Dimension n mindestens haben muss.
- Das Geschlechtssymbol eines positiv definiten Gitters quadratfreier Stufe bestimmen.
- Ein Gitter des `MAGMA`-Typs `LatNF` zum Typen `Lat` konvertieren.
- Testen ob ein Gitter L ℓ-modular ist.
- Testen ob ein Gitter L stark ℓ-modular ist.

```
load "hu.m"; // Program by David Lorch for constructing lattices with given elementary divisors

HermiteBounds := [1, 1.1547, 1.2599, 1.1412, 1.5157, 1.6654, 1.8115, 2, 2.1327, 2.2637, 2.3934, 2.5218, 2.6494, 2.7759, 2.9015, 3.0264, 3.1507, 3.2744, 3.3975, 3.5201, 3.6423, 3.7641, 3.8855, 4.0067, 4.1275, 4.2481, 4.3685, 4.4887, 4.6087, 4.7286, 4.8484, 4.9681, 5.0877, 5.2072, 5.3267, 5.4462];


function IdealConjugate(I, K)
// Input: Z_K-Ideal I; Field K

// Output: Z_K-Ideal which is the complex conugate of I

  gens := [];
  for g in Generators(I) do
    Append(~gens, ComplexConjugate(K ! g));
  end for;

  return ideal<Integers(K)|gens>;

end function;


function ReduceByIsometry(Lattices)
// Input: List of lattices

// Output: Reduced list for which the elements are pairwise non-isometric

  LatticesReduced := [* *];
  Minima := [* *];
  NumShortest := AssociativeArray();
  SizeAuto := AssociativeArray();


  for i in [1..#Lattices] do
    L := Lattices[i];

    min_computed := false;
    minimum := 0;

    shortest_computed := false;
    shortest := 0;

    auto_computed := false;
    auto := 0;

    for j in [1..#LatticesReduced] do
      M := LatticesReduced[j];

      if not min_computed then
        min_computed := true;
        minimum := Min(L);
```

```
      end if;

      if not IsDefined(Minima, j) then
        Minima[j] := Min(M);
      end if;

      if minimum ne Minima[j] then
        continue;
      end if;


      if not shortest_computed then
        shortest_computed := true;
        shortest := #ShortestVectors(L);
      end if;

      if not IsDefined(NumShortest, j) then
        NumShortest[j] := #ShortestVectors(M);
      end if;

      if shortest ne NumShortest[j] then
        continue;
      end if;


      if not auto_computed then
        auto_computed := true;
        auto := #AutomorphismGroup(L);
      end if;

      if not IsDefined(SizeAuto, j) then
        SizeAuto[j] := #AutomorphismGroup(M);
      end if;

      if auto ne SizeAuto[j] then
        continue;
      end if;


      if IsIsometric(L, M) then
        continue i;
      end if;
    end for;

    Append(~LatticesReduced, Lattices[i]);

    NewIndex := #LatticesReduced;
    if min_computed then
      Minima[NewIndex] := minimum;
    end if;

    if shortest_computed then
      NumShortest[NewIndex] := shortest;
    end if;

    if auto_computed then
```

```
      SizeAuto[NewIndex] := auto;
    end if;

  end for;

  return LatticesReduced;
end function;


function ExtremalMinimum(l, n)
// Input: Square-free l in N; n in N

// Output: Minimum that a l-modular lattice of dimension n must have at least

  if l eq 1 then k := 24;
  elif l eq 2 then k := 16;
  elif l eq 3 then k := 12;
  elif l eq 5 then k := 8;
  elif l eq 6 then k := 8;
  elif l eq 7 then k := 6;
  elif l eq 11 then k := 4;
  elif l eq 14 then k := 4;
  elif l eq 15 then k := 4;
  elif l eq 23 then k := 2;
  end if;

  return 2 + 2*Floor(n/k);
end function;


function GenSymbol(L)
// Input: Positive definite Numberfield Lattice L of square-free level

// Output: Genus symbol of L in the form [S_1, n, <2, n_2, epsilon_2, S_2, t_2>, <3, n_3, epsilon_3>, <5,...>, ...] for all primes dividing Det(L)
  Symbol := [* *];

  Rat := RationalsAsNumberField();
  Int := Integers(Rat);

  LNF := NumberFieldLatticeWithGram(Matrix(Rat, GramMatrix(L)));
  _, Grams2, Vals2 := JordanDecomposition(LNF,ideal<Int|2>);

  // Checks if all diagonal entries of the 1-component of the 2-adic jordan decomposition are even
  if Vals2[1] ne 0 or (Vals2[1] eq 0 and &and([Valuation(Rationals() ! (Grams2[1][i][i]), 2) ge 1 : i in [1..NumberOfRows(Grams2[1])]])) then
    Append(~Symbol, 2);
  else
    Append(~Symbol, 1);
  end if;
```

```
  Append(~Symbol, Dimension(L));


  for p in PrimeDivisors(Integers() ! (Determinant(L))) do
    _, Gramsp, Valsp := JordanDecomposition(LNF, ideal<Int|p>);

    if Valsp[1] eq 0 then
      G := Matrix(Rationals(), 1/p * Gramsp[2]);
    else
      G := Matrix(Rationals(), 1/p * Gramsp[1]);
    end if;

    sym := <p, NumberOfRows(G)>;

    det := Determinant(G);
    det := Integers() ! (det * Denominator(det)^2);

    if p eq 2 then
      if IsDivisibleBy(det+1, 8) or IsDivisibleBy(det-1, 8) then
        Append(~sym, 1);
      else
        Append(~sym, -1);
      end if;

      if &and([Valuation(Rationals() ! (G[i][i]), 2) ge 1 : i in [1..sym[2]]]) then
        Append(~sym, 2);
      else
        Append(~sym, 1);
      end if;

      if sym[4] eq 2 then
        Append(~sym, 0);
      else
        Append(~sym, Integers() ! (Trace(G)* Denominator(Trace(G))^2) mod 8);
      end if;
    else
      Append(~sym, LegendreSymbol(det, p));
    end if;

    Append(~Symbol, sym);
  end for;

  return Symbol;

end function;


function ToZLattice(L)
// Input: Numberfield lattice L

// Output: L as Z-lattice
  B:= Matrix(ZBasis(L`Module));
```

```
  G:= B * L`Form * InternalConjugate(L, B);
  Form:= Matrix( Ncols(G), [ AbsoluteTrace(e) : e in Eltseq(G) ] );
  Form:=IntegralMatrix(Form);

  LZ := LatticeWithGram(LLLGram(Matrix(Integers(),Form)));

  return LZ;
end function;


function MiPoQuotient(sigma, L, p);
// Input : Automorphism sigma of L; Lattice L

// Output: Minimal polynomial of the operation of sigma on the partial dual quotient L^(#, p) / L

  sigma := Matrix(Rationals(), sigma);
  L := CoordinateLattice(L);
  LD := PartialDual(L, p : Rescale := false);
  _,phi := LD / L;
  MiPo := PolynomialRing(GF(p)) ! 1;

  B := [];

  for i in [1..Rank(LD)] do

    b := LD.i;
    if b in sub<LD|L,B> then
      continue;
    end if;
    Append(~B,b);

    dep := false;
    C := [Eltseq(phi(b))];
    while not dep do
      b := b*sigma;
      Append(~C, Eltseq(phi(b)));
      Mat := Matrix(GF(p),C);
      if Dimension(Kernel(Mat)) gt 0 then
        dep := true;
        coeff := Basis(Kernel(Mat))[1];
        coeff /:= coeff[#C];
        coeff := Eltseq(coeff);
        MiPo := LCM(MiPo, Polynomial(GF(p), coeff));
      else
        Append(~B, b);
      end if;
    end while;
  end for;

  return MiPo;

end function;

function IsModular(L, l)
```

```
// Input: Lattice L; l in N

// Output: true iff L is a l-modular lattice

  return IsIsometric(L, LatticeWithGram(l*GramMatrix(Dual
(L:Rescale:=false))));

end function;

function IsStronglyModular(L,l)
// Input: Lattice L; l in N

// Output: true iff L is a strongly l-modular lattice

  return &and[IsIsometric(L, LatticeWithGram(m*GramMatrix
(PartialDual(L, m : Rescale:=false)))) : m in [m : m in Divisors
(l) | Gcd(m, Integers() ! (l/m)) eq 1]];

end function;
```

§ 6.3 `MAGMA`-Implementierungen der Ideal-Gitter-Algorithmen

Es folgt der Quellcode zu den Algorithmen aus Kapitel 3. Die implementierten Funktionen sind in der Reihenfolge des Quellcodes:

- Alle Teiler eines $\mathbb{Z}_K$-Ideals $\mathcal{I}$ von festgelegter Norm berechnen. Siehe Algorithmus (1).
- Einen total-reellen Erzeuger eines $\mathbb{Z}_K$-Ideals $\mathcal{I}$ bestimmen. Siehe Algorithmus (2).
- Matrix, deren Einträge die Vorzeichen der reellen Einbettung der Grundeinheiten kodieren sowie eine Liste aller total-positiven Elemente in $\mathbb{Z}_{K+}^*$ reduziert nach $\{\lambda\overline{\lambda} \mid \lambda \in \mathbb{Z}_K^*\}$ und eine Liste von Erzeugern einer Untergruppe von $\mathbb{Z}_{K+}^*$ mit ungeradem Index bestimmen. Siehe Algorithmus (3)
- Das Minimalpolynom der Operation eines Automorphismus σ von einem Gitter L auf dem $\mathbb{F}_p$-Vektorraum $L^{\#,p}/L$ berechnen
- Liste aller total-positiven Erzeuger eines $\mathbb{Z}_K$-Ideals $\mathcal{I}$ reduziert nach $\{\lambda\overline{\lambda} \mid \lambda \in \mathbb{Z}_K^*\}$ bestimmen. Siehe ebenfalls Algorithmus (3).
- Liste von Vertretern der Klassengruppe von K modulo der Operation der Galoisgruppe $\mathrm{Gal}(K/\mathbb{Q})$ bestimmen. Siehe Abschnitt (3.3).
- Aus einem $\mathbb{Z}_K$-Ideal $\mathcal{I}$ und einem total-positiven Element α das Gitter, welches durch Auffassung von $\mathcal{I}$ als $\mathbb{Z}$-Gitter mit Bilinearform $b(x, y) := \mathrm{Spur}(\alpha x\overline{y})$ entsteht.
- Alle Ideal-Gittern über gegebenem CM-Körper mit vorgegebener Determinante und quadratfreier Stufe aufzählen. Siehe Algorithmus (4).

- Liste von allen ℓ-modularen Gittern in Dimension n erstellen, welche einen Automorphismus σ besitzen mit $\mu_\sigma = \Phi_m$ und $\varphi(m) = n$. Siehe Abschnitt (3.5).
- Schleife, welche die letzte Funktion für verschiedene n und ℓ auswertet und die Ergebnisse speichert.

```
load "Utility.m";

function DivisorsWithNorm(I, n)
// Input: Z_K-Ideal I; norm n in Z

// Output: List of divisors of I with norm n

  norm := Integers() ! Norm(I);

  if n eq 1 then return [I*I^(-1)]; end if;
  if not IsDivisibleBy(norm, n) then return []; end if;
  if norm eq n then return [I]; end if;

  Fact := Factorization(I);

  p1 := Fact[1][1];
  s1 := Fact[1][2];
  np := Integers() ! Norm(p1);

  Results := [];

  for j in [0..s1] do
    if IsDivisibleBy(n, np^j) then
      B := DivisorsWithNorm(I*p1^(-s1), Integers() ! (n / np^j));

      for J in B do
        Append(~Results, p1^j*J);
      end for;
    end if;
  end for;

  return Results;

end function;


function TotallyRealGenerator(I, K, Kpos)
// Input: Z_K-Ideal I; Field K; Field Kpos

// Output: Boolean that indicates success; totally real generator of I cap Kpos

  ZK := Integers(K);
  ZKpos := Integers(Kpos);

  Ipos:=ideal<ZKpos|1>;
  Split:=[];

  Fact := Factorization(I);

  for i in [1..#Fact] do
    if i in Split then continue; end if;

    pi:=Fact[i][1];
    si:=Fact[i][2];
```

```
    piConj := IdealConjugate(pi,K);

    p:=MinimalInteger(pi);

    pFact:=Factorization(ideal< ZKpos | p >);

    for qj in [fact[1] : fact in pFact] do
      if ideal<ZK | Generators(qj)> subset pi then
        a := qj;
        break;
      end if;
    end for;

    aZK := ideal<ZK|Generators(a)>;

    if aZK eq pi^2 then

      if not IsDivisibleBy(si, 2) then return false, _; end if;
      Ipos *:= a^(Integers() ! (si/2));

    elif aZK eq pi then

      Ipos *:= a^si;

    elif aZK eq pi*piConj then

      if Valuation(I, pi) ne Valuation(I, piConj) then return false, _; end if;
      Ipos *:= a^si;
      for j in [1..#Fact] do
        pj := Fact[j][1];
        if pj eq piConj then
          Append(~Split, j);
          break;
        end if;
      end for;
    end if;
  end for;

  return IsPrincipal(Ipos);

end function;


function EmbeddingMatrix(K, Kpos)
// Input: Field K; Field Kpos

// Output: Matrix M whose entries give the signs of the embeddings of the fundamental units; List U of all totally positive units in ZKpos modulo norms; List of generators of a subgroup of Z_Kpos^* of odd index

  ZKpos := Integers(Kpos);

  t := #Basis(ZKpos);

```

```
  G, mG := pFundamentalUnits(ZKpos,2);
  FundUnits := [mG(G.i) : i in [1..t]];

  M := ZeroMatrix(GF(2), t, t);

  for i in [1..t] do
    Embeds := RealEmbeddings(FundUnits[i]);
    for j in [1..t] do
      if Embeds[j] lt 0 then
        M[i][j] := 1;
      end if;
    end for;
  end for;

  U := [];
  for a in Kernel(M) do
    e := ZKpos ! &*[FundUnits[i]^(Integers() ! a[i]) : i in [1..t]];
    Append(~U, e);
  end for;

  ZRel := Integers(RelativeField(Kpos, K));

  Units := [];
  for u in U do
    for w in Units do
      if NormEquation(ZRel,ZRel ! (u/w)) then
        continue u;
      end if;
    end for;

    Append(~Units, u);
  end for;

  return M, Units, FundUnits;

end function;


function TotallyPositiveGenerators(alpha, K, Kpos, M, U, FundUnits)
// Input: alpha in ZKpos; Field K; Field Kpos; Embedding-Matrix M; List U of all totally-positive units in ZKpos modulo norms; List FundUnits of generators of a subgroup of Z_Kpos^* of odd index

// Output: Boolean that inducates success; List of all totally-positive generators of alpha*ZK modulo norms

  t := #Basis(Kpos);
  V := ZeroMatrix(GF(2), 1, t);

  Embeds := RealEmbeddings(alpha);
  for i in [1..t] do
    if Embeds[i] lt 0 then
      V[1][i] := 1;
```

```
      end if;
    end for;

    solvable, x := IsConsistent(M,V);
    if not solvable then
      return false, _;
    end if;

    g := Integers(Kpos) ! &*[FundUnits[i]^(Integers() ! x[1][i]) : i in [1..t]];

    return true, [alpha*g*u : u in U];

end function;


function ClassesModGalois(K)
// Input : Field K

// Output : List of representatives of the class group of Z_K modulo the action of the Galois-group of K/Q

    ZK := Integers(K);
    Cl, mCl := ClassGroup(ZK : Proof:="GRH");

    ClModGal:=[];
    for a in Cl do
      A:=mCl(a);
      for f in Automorphisms(K) do
        if Inverse(mCl)(ideal<ZK | [f(x) : x in Generators(A)]>) in ClModGal then
          continue a;
        end if;
      end for;
      Append(~ClModGal,a);
    end for;

    return [mCl(g) : g in ClModGal];

end function;



function LatFromIdeal(J, alpha, K)
// Input: ZK-Ideal J; Totally positive element alpha Kpos; Field K

// Output: Z-Lattice with elements J and inner product (x,y) := Tr(alpha*x*Conj(y))

    n := #Basis(K);
    z := PrimitiveElement(K);

    GeneratorMatrix := KMatrixSpace(Rationals(), #Generators(J)*n, n) ! 0;

```

```
  for i in [1..#Generators(J)] do
    g := K ! (Generators(J)[i]);

    for j in [1..n] do
      GeneratorMatrix[(i-1)*n + j] := Vector(Rationals(), n, Eltseq(g*z^(j-1)));
    end for;
  end for;


  BaseVecs := Basis(Lattice(GeneratorMatrix));

  ZBase := [];
  for i in [1..n] do
    b := K ! 0;
    for j in [1..n] do
      b +:= BaseVecs[i][j]*z^(j-1);
    end for;
    Append(~ZBase, b);
  end for;

  InnProd := KMatrixSpace(Rationals(), n, n) ! 0;
  for i in [1..n] do
    for j in [1..n] do
      InnProd[i][j] := Trace(K ! (alpha * z^(i-j)));
    end for;
  end for;

  L := LatticeWithBasis(KMatrixSpace(Rationals(), n, n) ! Matrix(BaseVecs), InnProd);
  L := LatticeWithGram(LLLGram(GramMatrix(L)));

  return L;

end function;


function IdealLattices(d, K, Kpos, A, M, U, FundUnits, Reduce)
// Input: d in N; Field K; Field Kpos; Class Group of K mod Galois-Group A; Embedding-Matrix M; List of totally-positive units U; List FundUnits of generators of a subgroup of Z_Kpos^* of odd index; Boolean Reduce that indicates, whether the list shall be reduced by isometry.

// Output: List of all even ideal-lattices over K of square-free level and determinant d

  ZK := Integers(K);
  InvDiff := Different(ZK)^(-1);

  l := &*(PrimeDivisors(d));

  B := DivisorsWithNorm(ideal<ZK|l>, d);

  Results := [];

```

```
  for I in A do
    for b in B do
      J := (I*IdealConjugate(I,K))^(-1)*InvDiff*b;

      x, alphaPrime := TotallyRealGenerator(J, K, Kpos);

      if x then
        y, TotPos := TotallyPositiveGenerators(alphaPrime, K, Kpos, M, U, FundUnits);
        if y then
          for alpha in TotPos do
            L := LatFromIdeal(I, alpha, K);
            if IsEven(L) then
              Append(~Results, L);
            end if;
          end for;
        end if;
      end if;
    end for;
  end for;

  if Reduce then Results := ReduceByIsometry(Results); end if;

  return Results;

end function;


function ModIdLat(l, n)
// Input: square-free l in N; n in N

// Output: List of all l-modular lattices of dimension n that are ideal lattices over some cyclotomic field reduced by isometry

  det := l^(Integers() ! (n/2));

  Lattices := [];

  for m in [m : m in EulerPhiInverse(n) | m mod 4 ne 2] do
    K<z> := CyclotomicField(m);
    Kpos := sub<K | z + z^(-1)>;

    A := ClassesModGalois(K);
    M, U, FundUnits := EmbeddingMatrix(K, Kpos);
    Lattices cat:= IdealLattices(det, K, Kpos, A, M, U, FundUnits, false);
  end for;

  Lattices := ReduceByIsometry(Lattices);

  PrintFileMagma(Sprintf("IdealLattices/%o-Modular/%o-Dimensional", l, n), Lattices : Overwrite := true);

  return Lattices;

```

```
end function;

procedure MainLoop()
  for n := 2 to 36 by 2 do
    for l in [1,2,3,5,6,7,11,14,15,23] do
      printf "dim = %o, l = %o\n", n, l;
      Results := ModIdLat(l, n);
      ModList := [L : L in Results | IsModular(L, l)];
      StrongModList := [L : L in Results | IsStronglyModular(L,l)];
      PrintFileMagma(Sprintf("SubidealLattices/%o-Modular/%o-Dimensional", l, n), Results : Overwrite := true);
      PrintFileMagma(Sprintf("SubidealLattices/%o-Modular/%o-DimensionalModular", l, n), ModList : Overwrite := true);
      PrintFileMagma(Sprintf("SubidealLattices/%o-Modular/%o-DimensionalStronglyModular", l, n), StrongModList : Overwrite := true);

      if #Results gt 0 then
        printf "\n\n----------n = %o, l = %o: %o lattices found! %o of them are modular and %o are strongly modular----------\n\n", n, l, #Results, #ModList, #StrongModList;
      end if;
    end for;
  end for;
end procedure;
```

§ 6.4 `MAGMA`-Implementierungen der Subideal-Gitter-Algorithmen

Es folgt der Quellcode zu den Algorithmen aus Kapitel 4 bis einschließlich Abschnitt (4.6). Die implementierten Funktionen sind in der Reihenfolge des Quellcodes:

- Mithilfe von Hermite-Schranken die möglichen Typen von Automorphismen von Primzahlordnung gerader Gitter mit quadratfreier Stufe ℓ, Determinante ℓ^k und Minimum $\geq t$ aufzählen. Siehe Algorithmus (5).

- Mit dem Kneser'schen Nachbarverfahren die Vertreter der Isometrieklassen aller Gitter in dem Geschlecht eines Vertreters bestimmen. Siehe Algorithmus (6).

- Anhand von vorgegebener Dimension und Determinante Vertreter aller Isometrieklassen von (wenn gewünscht geraden) Gittern mit dieser Dimension und Determinante und mit quadratfreier Stufe aufzählen.

- Zu einem vorgegebenen Geschlechtssymbol Vertreter aller Isometrieklassen von Gittern mit diesem Geschlechtssymbol und quadratfreier Stufe aufzählen.

- Mit der `MAGMA`-Methode `Sublattices` σ-invariante Obergitter von Index p^s für einen Gitterautomorphismus σ bestimmen.

- Mit der in Abschnitt (4.5) beschriebenen Methode Obergitter von $L_1 \perp L_p$ invariant unter $\mathrm{diag}(\sigma_1, \sigma_p)$ bestimmen. Siehe Algorithmus (8).

- Mit der Methode von Michael Jürgens alle Obergitter von L mit Index p^s bestimmen.

- Minimalpolynom der Operation eines Automorphismus σ von einem Gitter L auf dem $\mathbb{F}_p$-Vektorraum $L^{\#,p}/L$ bestimmen. Siehe Abschnitt (4.6).

- Liste aller extremalen ℓ-modularen Gitter in Dimension n bestimmen, die einen großen Automorphismus der Ordnung m haben, sodass m einen Primteiler p mit $\mathrm{ggT}(p, \ell) = 1$ und $\frac{\mu_\sigma}{\Phi_m} | (X^{\frac{m}{p}} - 1)$ besitzt. Siehe Algorithmus (9).
- Schleife, welche die letzte Funktion für verschiedene n und ℓ auswertet und die Ergebnisse speichert.

```
load "Utility.m";
load "IdealLattices.m";


function AutomorphismTypes(l, k, n, t)
// Input: Square-free l in N, k in N, n in N, t in N

// Output: List of all possible types of automorphisms of prime order for even lattices of level l with determinant l^k, dimension n and minimum greater or equal to t
  Results := [];

  lFactors := PrimeDivisors(l);

  for p in PrimesUpTo(n+1) do
    if p in lFactors then continue; end if;

    K<z> := CyclotomicField(p);
    Kpos := sub<K|z+1/z>;

    f := [];

    for q in lFactors do
      if p le 3 then
        Append(~f, 1);
      else
        Append(~f, InertiaDegree(Factorization(ideal<Integers(Kpos) | q>)[1][1]));
      end if;
    end for;

    for np in [i*(p-1) : i in [1..Floor(n/(p-1))]] do

      n1 := n - np;
      for s in [0..Min(n1, Integers() ! (np/(p-1)))] do
        if not IsDivisibleBy(s - Integers() ! (np / (p-1)), 2) then continue s; end if;
        if p eq 2 and not IsDivisibleBy(s, 2) then continue s; end if;

        if l eq 1 then
          if n1 gt 0 then
            Gamma1 := t/p^(s/n1);
            if Gamma1 gt HermiteBounds[n1] + 0.1 then continue; end if;
          end if;

          if np gt 0 then
            Gammap := t/p^(s/np);
            if Gammap gt HermiteBounds[np] + 0.1 then continue; end if;
          end if;
          type := <p, n1, np, s>;

          Append(~Results, type);
        else
```

```
            for kp in CartesianProduct([[2*f[i]*j : j in [0..Floor(Min(np,k)/(2*f[i]))]] : i in [1..#f]]) do

              k1 := [k - kp[i] : i in [1..#kp]];

              for i in [1..#kp] do
                if k1[i] gt Min(n1,k) then continue kp; end if;
                if not IsDivisibleBy(k1[i] - k, 2) then continue kp; end if;
                if not IsDivisibleBy(kp[i], 2) then continue kp; end if;
              end for;

              if n1 gt 0 then
                Gamma1 := p^s;
                for i in [1..#lFactors] do
                  Gamma1 *:= lFactors[i]^k1[i];
                end for;
                Gamma1 := t / Gamma1^(1/n1);

                if Gamma1 gt HermiteBounds[n1] + 0.1 then continue; end if;
              end if;

              if np gt 0 then
                Gammap := p^s;
                for i in [1..#lFactors] do
                  Gammap *:= lFactors[i]^kp[i];
                end for;
                Gammap := t / Gammap^(1/np);

                if Gammap gt HermiteBounds[np] + 0.1 then continue; end if;
              end if;

              if p eq 2 then
                if n1 gt 0 then
                  Gamma1 := 1;
                  for i in [1..#lFactors] do
                    Gamma1 *:= lFactors[i]^k1[i];
                  end for;
                  Gamma1 := t/2 / Gamma1^(1/n1);

                  if Gamma1 gt HermiteBounds[n1] + 0.1 then continue; end if;
                end if;

                if np gt 0 then
                  Gammap := 1;
                  for i in [1..#lFactors] do
                    Gammap *:= lFactors[i]^kp[i];
                  end for;
                  Gammap := t/2 / Gammap^(1/np);

                  if Gammap gt HermiteBounds[np] + 0.1 then continue; end if;
```

```
              end if;
            end if;

            type := <p, n1, np, s>;
            for i in [1..#lFactors] do
              Append(~type, lFactors[i]);
              Append(~type, k1[i]);
              Append(~type, kp[i]);
            end for;

            Append(~Results, type);
          end for;
        end if;
      end for;
    end for;
  end for;

  return Results;

end function;


function EnumerateGenusOfRepresentative(L)
// Input: Lattice L, t in N

// Output: List of all representatives of isometry-classes in the genus of L

  "Enumerate genus of representative";
  try return eval Read(Sprintf("GenusSymbols/Gen_%o", GenSymbol(L))); catch e; end try;

  if Dimension(L) le 6 then
    Gen := GenusRepresentatives(L);
    ZGen := [];
    for M in Gen do
      if Type(M) eq Lat then
        Append(~ZGen,LLL(M));
      else
        Append(~ZGen, LatticeWithGram(LLLGram(Matrix(Rationals(), GramMatrix(SimpleLattice(M))))));
      end if;
    end for;
    PrintFileMagma(Sprintf("GenusSymbols/Gen_%o",GenSymbol(L)), ZGen : Overwrite := true);
    return ZGen;
  end if;

  M := Mass(L);
  Gen := [L];
  Explored := [false];
  NumFound := [1];
  Minima := [Minimum(L)];
    NumShortest := [#ShortestVectors(L)];
    SizeAuto := [#AutomorphismGroup(L)];
  m := 1 / SizeAuto[1];
```

```
  p := 2;

  t0 := Realtime();

  while m lt M do
    //printf "So far %o classes found. Difference to actual mass is %o. \n", #Gen, M-m;
    if Realtime(t0) ge 120*60 then
      printf "2 hours have elapsed and not the whole genus was explored. Remaining difference to actual mass is %o. %o classes were found so far. The symbol is %o.\n", M-m, #Gen, GenSymbol(L);
      return Gen;
    end if;

    RareFound := [];
    MinCount := Infinity();

    if &and(Explored) then
      "All explored. Going to next prime.";
      Explored := [false : x in Explored];
      p := NextPrime(p);
      if p ge 5 and Dimension(L) ge 8 then
        printf "Prime too large, cannot continue constructing neighbours. Remaining difference to actual mass is %o. %o classes were found so far. The symbol is %o.\n", M-m, #Gen, GenSymbol(L);
        return Gen;
      end if;
      if p ge 3 and Dimension(L) ge 12 then
        printf "Prime too large, cannot continue constructing neighbours. Remaining difference to actual mass is %o. %o classes were found so far. The symbol is %o.\n", M-m, #Gen, GenSymbol(L);
        return Gen;
      end if;
    end if;

    for i in [1..#Gen] do
      if not Explored[i] then
        if NumFound[i] lt MinCount then
          RareFound := [i];
          MinCount := NumFound[i];
        elif NumFound[i] eq MinCount then
          Append(~RareFound, i);
        end if;
      end if;
    end for;

    i := RareFound[Random(1, #RareFound)];

    Neigh := [CoordinateLattice(N) : N in Neighbours(Gen[i], p)];
    Explored[i] := true;

```

```
    for N in Neigh do

          auto := #AutomorphismGroup(N);
          if auto lt 1/(M-m) then continue; end if;

          minimum := Minimum(N);
          shortest := #ShortestVectors(N);

          for j in [1..#Gen] do

              if minimum ne Minima[j] then
                  continue j;
              end if;

              if shortest ne NumShortest[j] then
                  continue j;
              end if;

              if auto ne SizeAuto[j] then
                  continue j;
              end if;

              if IsIsometric(N, Gen[j]) then
               NumFound[j] +:= 1;
               continue N;
              end if;
          end for;

        Append(~Gen,N);
        Append(~Explored, false);
        Append(~NumFound, 1);
        Append(~Minima, minimum);
        Append(~NumShortest, shortest);
        Append(~SizeAuto, auto);
        m +:= 1/auto;
        if m eq M then
          break N;
        end if;
        end for;
    end while;

    PrintFileMagma(Sprintf("GenusSymbols/Gen_%o",GenSymbol(L)), Gen : Overwrite := true);

    return Gen;

end function;


function EnumerateGenusDeterminant(det, n, even)
// Input: det in N; n in N; boolean even that indicates whether only even lattices shall be enumerated

// Output: Representatives of all isometry-classes belonging to a genus of integral lattices with determinant det, dimension n, and square-free level
```

```
  if n eq 0 then
    return [LatticeWithGram(Matrix(Rationals(),0,0,[]))];
  end if;

  if n eq 1 then
    L := LatticeWithGram(Matrix(Rationals(), 1, 1, [det]));
    Symbol := GenSymbol(L);
    if even and not Symbol[1] eq 2 then return []; end if;
    if not IsSquarefree(Level(L)) then return []; end if;
    if even and IsDivisibleBy(Determinant(L), 2) then
      if not Symbol[3][4] eq 2 then return []; end if;
    end if;
    return [L];
  end if;

  if n eq 2 then
    Results := [];

    for m in [1..Floor(1.155*Sqrt(det))] do
      for a in [-m+1..m-1] do

        if not IsDivisibleBy(det + a^2, m) then continue; end if;
        b := Integers() ! ((det + a^2) / m);

        if b lt m then continue; end if;
        if even and not IsEven(b) then continue; end if;

        Mat := Matrix(Rationals(), 2, 2, [m,a,a,b]);
        if not IsPositiveDefinite(Mat) then continue; end if;

        L := LatticeWithGram(Mat);

        if not IsSquarefree(Level(L)) then continue; end if;

        Symbol := GenSymbol(L);
        if even and not Symbol[1] eq 2 then continue; end if;
        if even and IsDivisibleBy(Determinant(L), 2) then
          if not Symbol[3][4] eq 2 then continue; end if;
        end if;

        Append(~Results, L);
      end for;
    end for;

    return ReduceByIsometry(Results);
  end if;


  Rat := RationalsAsNumberField();
  Int := Integers(Rat);

  primes := PrimeBasis(det);
  exps := [Valuation(det, p) : p in primes];

```

```
  IdealList := [];
  if not 2 in primes then
    Append(~IdealList, <ideal<Int|2>, [[0,n]]>);
  end if;

  for i in [1..#primes] do
    p := primes[i];
    e := Abs(exps[i]);
    if n eq e then
      Append(~IdealList, <ideal<Int|p>, [[1,e]]>);
    elif e eq 0 then
      Append(~IdealList, <ideal<Int|p>, [[0,n]]>);
    else
      Append(~IdealList, <ideal<Int|p>, [[0,n-e],[1,e]]>);
    end if;
  end for;

  "Constructing representatives";
  try
    Rep := LatticesWithGivenElementaryDivisors(Rat, n, IdealList);
  catch e
    print "Error while trying to construct a representative. IdealList:";
    IdealList;
    return [];
  end try;

  Results := [];

  for L in Rep do

    LZ := ToZLattice(L);
    if IsSquarefree(Level(LZ)) then
      Symbol := GenSymbol(LZ);
      if even and not Symbol[1] eq 2 then continue L; end if;
      if even and IsDivisibleBy(det, 2) then
        if not Symbol[3][4] eq 2 then continue L; end if;
      end if;

      Gen := EnumerateGenusOfRepresentative(LZ);
      Results cat:= Gen;
    end if;
  end for;

  return Results;

end function;


function EnumerateGenusSymbol(Symbol)
// Input: Genus-symbol Symbol of positive definite lattices of square-free level; t in N

// Output: Representatives of all isometry-classes belonging to the genus
```

```
  try return eval Read(Sprintf("GenusSymbols/Gen_%o", Symbol));
catch e; end try;

  n := Symbol[2];

  if n eq 0 then
    return [LatticeWithGram(Matrix(Rationals(),0,0,[]))];
  end if;

  if n eq 1 then
    det := &*[Symbol[i][1]^Symbol[i][2] : i in [3..#Symbol]];
    L := LatticeWithGram(Matrix(Rationals(), 1, 1, [det]));
    if GenSymbol(L) eq Symbol then
      return [L];
    end if;
    return [];
  end if;

  if n eq 2 then
    det := &*[Symbol[i][1]^Symbol[i][2] : i in [3..#Symbol]];

    for m := 2 to Floor(1.155*Sqrt(det)) by 2 do
      for a in [-m+1..m-1] do

        if not IsDivisibleBy(det + a^2, m) then continue; end if;
        b := Integers() ! ((det + a^2) / m);

        if b lt m then continue; end if;
        if not IsEven(b) then continue; end if;

        Mat := Matrix(Rationals(), 2, 2, [m,a,a,b]);
        if not IsPositiveDefinite(Mat) then continue; end if;

        L := LatticeWithGram(Mat);

        if not IsSquarefree(Level(L)) then continue; end if;

        if Symbol eq GenSymbol(L) then
          return EnumerateGenusOfRepresentative(L);
        end if;
      end for;
    end for;

    return [];

  end if;

  Rat := RationalsAsNumberField();
  Int := Integers(Rat);

  IdealList := [];
  if Symbol[3][1] ne 2 then
    Append(~IdealList, <ideal<Int|2>, [[0,n]]>);
  end if;
```

```

  for i in [3..#Symbol] do
    p := Symbol[i][1];
    np := Symbol[i][2];

    if n eq np then
      Append(~IdealList, <ideal<Int|p>, [[1,np]]>);
    elif np eq 0 then
      Append(~IdealList, <ideal<Int|p>, [[0,n]]>);
    else
      Append(~IdealList, <ideal<Int|p>, [[0,n-np],[1,np]]>);
    end if;
  end for;

  "Constructing representatives";
  try
    Rep := LatticesWithGivenElementaryDivisors(Rat, n, IdealList);
  catch e
    print "Error while trying to construct a representative. IdealList:";
    IdealList;
    return [];
  end try;

  for L in Rep do
    LZ := ToZLattice(L);
    if GenSymbol(LZ) eq Symbol then
      Gen := EnumerateGenusOfRepresentative(LZ);
      return Gen;
    end if;
  end for;

  return [];

end function;


function SuperLatticesMagma(L, p, s, sigma)
// Input: Lattice L; Prime p; s in N; Automorphism sigma of L

// Output: All even sigma-invariant superlattices of L with index p^s using magmas method


  LD := PartialDual(L,p:Rescale:=false);

  G := MatrixGroup<NumberOfRows(sigma), Integers() | sigma >;
  den1 := Denominator(BasisMatrix(LD));
  den2 := Denominator(InnerProductMatrix(LD));

  A := LatticeWithBasis(G, Matrix(Integers(), den1*BasisMatrix(LD)), Matrix(Integers(), den2^2*InnerProductMatrix(LD)));

  SU := [];
  SU := Sublattices(A, p : Levels := s, Limit := 100000);
```

```

  if #SU eq 100000 then "List of sublattices is probably not complete."; end if;

  Results := [];

  for S in SU do

    M := 1/den1 * 1/den2 * S;

    if Determinant(M)*p^(2*s) eq Determinant(L) then
      Append(~Results, M);
    end if;
  end for;

  return [L : L in Results | IsEven(L)];
end function;


function SuperLattices(L1, Lp, p, sigma1, sigmap)
// Input: Lattice L1; Lattice Lp; Prime p; Automorphism sigma of L

// Output: All even superlattices of L1 + Lp invariant under diag (sigma1, sigmap) with index p^s using isometry-method

  M := OrthogonalSum(L1, Lp);

  L1Quot, phi1 := PartialDual(L1,p : Rescale:=false) / L1;
  LpQuot, phip := PartialDual(Lp,p : Rescale:=false) / Lp;

  m := #Generators(L1Quot);

  phi1Inv := Inverse(phi1);
  phipInv := Inverse(phip);

  G1 := ZeroMatrix(GF(p),m,m);
  Gp := ZeroMatrix(GF(p),m,m);
  for i in [1..m] do
    for j in [1..m] do
      G1[i,j] := GF(p) ! (p*InnerProduct(phi1Inv(L1Quot.i), phi1Inv(L1Quot.j)));
      Gp[i,j] := GF(p) ! (-p*InnerProduct(phipInv(LpQuot.i), phipInv(LpQuot.j)));
    end for;
  end for;

  V1 := KSpace(GF(p), m, G1);
  Vp := KSpace(GF(p), m, Gp);

  O1 := IsometryGroup(V1);

  sigma1Quot := ZeroMatrix(GF(p),m,m);
  for i in [1..m] do
    sigma1Quot[i] := Vector(GF(p), Eltseq(phi1(phi1Inv
```

```
(L1Quot.i)*Matrix(Rationals(),sigma1))));
  end for;

  sigmapQuot := ZeroMatrix(GF(p),m,m);
  for i in [1..m]
do
    sigmapQuot[i] := Vector(GF(p), Eltseq(phip(phipInv
(LpQuot.i)*Matrix(Rationals(),sigmap))));
  end for;

  CL1Quot := Centralizer(O1, O1 ! sigma1Quot);

  CL1 := Centralizer(AutomorphismGroup(L1), sigma1);

  CL1ProjGens := [];
  for g in Generators(CL1) do
    gProj := ZeroMatrix(GF(p),m,m);
    for i in [1..m] do
      gProj[i] := Vector(GF(p), Eltseq(phi1(phi1Inv
(L1Quot.i)*Matrix(Rationals(), g))));
    end for;
    Append(~CL1ProjGens, gProj);
  end for;

  CL1Proj := MatrixGroup<m, GF(p) | CL1ProjGens>;

  _, psi := IsIsometric(V1,Vp);

  psi := MatrixOfIsomorphism(psi);
  _, u := IsConjugate(O1, O1 ! sigma1Quot, O1 !
(psi*sigmapQuot*psi^(-1)));

  phi0 := u*psi;

  U, mapU := CL1Quot / CL1Proj;

  LphiList := [];
  for u in U do
    phi := Inverse(mapU)(u)*phi0;

    Gens := [];
    for i in [1..m] do
      x := phi1Inv(L1Quot.i);
      y := phipInv(LpQuot ! Eltseq(phi[i]));
      Append(~Gens, Eltseq(x) cat Eltseq(y));
    end for;

    Lphi := ext<M | Gens>;
    Append(~LphiList,LatticeWithGram(LLLGram(GramMatrix(Lphi))));
  end for;

  return [L : L in LphiList | IsEven(L)];
end function;
```

```
function SuperLatticesJuergens(L, p, s)
// Input: Lattice L; Prime p; s in N; t in N

// Output: All even superlattices of L with index p^s using juergens method

  if s eq 0 then
    return [L];
  end if;

  T,mapT:=PartialDual(L,p:Rescale:=false) / L;
  mapTinv := Inverse(mapT);

    m:=#Generators(T);
    G:=GramMatrix(L);
    G_F:=MatrixAlgebra(GF(p),m)!0;

    for i:=1 to m do
        for j:=1 to m do
            G_F[i,j]:=GF(p)!(p*InnerProduct(mapTinv(T.i),mapTinv(T.j)));
        end for;
    end for;

    V:=KSpace(GF(p),m,G_F);
    if not s le WittIndex(V) then
        return [];
    end if;

    M1:=MaximalTotallyIsotropicSubspace(V);
    M:=sub< M1 | Basis(M1)[1..s] >;

    O:=IsometryGroup(V);
    Aut:=AutomorphismGroup(L:Decomposition:=true);

    Gens:=[];
    for g in Generators(Aut) do
        g_F:=MatrixAlgebra(GF(p),m)!0;
        for i:=1 to m do
            g_F[i]:=V!Vector(Eltseq(mapT(mapTinv(T!Eltseq(V.i))*Matrix(Rationals(),g))));
        end for;
        Append(~Gens,g_F);
    end for;

    O_L:=sub< O | Gens>;
    mapS,S,Kernel:=OrbitAction(O_L,Orbit(O,M));
    Set:=[Inverse(mapS)(i[2]) : i in OrbitRepresentatives(S)];
    SuperLat := [CoordinateLattice(ext< L | [mapTinv(T!Eltseq(x)) : x in Basis(W)] >) : W in Set];

    return [L : L in SuperLat | IsEven(L)];

end function;

```

```
function MiPoQuotient(sigma, L, p);
// Input : Automorphism sigma of L; Lattice L

// Output: Minimal polynomial of the operation of sigma on the partial dual quotient L^(#, p) / L

    sigma := Matrix(Rationals(), sigma);
    L := CoordinateLattice(L);
    LD := PartialDual(L, p : Rescale := false);
    _,phi := LD / L;
    MiPo := PolynomialRing(GF(p)) ! 1;

    B := [];

    for i in [1..Rank(LD)] do

        b := LD.i;
        if b in sub<LD|L,B> then
            continue;
        end if;
        Append(~B,b);

        dep := false;
        C := [Eltseq(phi(b))];
        while not dep do
            b := b*sigma;
            Append(~C, Eltseq(phi(b)));
            Mat := Matrix(GF(p),C);
            if Dimension(Kernel(Mat)) gt 0 then
                dep := true;
                coeff := Basis(Kernel(Mat))[1];
                coeff /:= coeff[#C];
                coeff := Eltseq(coeff);
                MiPo := LCM(MiPo, Polynomial(GF(p), coeff));
            else
                Append(~B, b);
            end if;
        end while;
    end for;

    return MiPo;

end function;


function ConstructLattices(l, n)
// Input: Square-free l; n in N

// Output: List of all extremal l-modular lattices that have a large automorphism sigma of order m, such that there is a prime divisor p of m with ggT(p,l) = 1 and mu_sigma / Phi_m | (x^(m/p) - 1)
  Results := [];

  min := ExtremalMinimum(l,n);
```

```

  AutoTypes := AutomorphismTypes(l, Integers() ! (n/2), n, min);

  for phim in [Integers() ! (n/2)+1 .. n] do

    n1 := n - phim;
    np := phim;

    for m in EulerPhiInverse(phim) do

      printf "m = %o\n", m;

      for p in PrimeDivisors(m) do
        //printf "Testing p = %o\n", p;
        if Gcd(p,l) ne 1 then continue; end if;
        d := Integers() ! (m/p);
        PossibleTypes := [type : type in AutoTypes | type[1] eq p and type[2] eq n1 and type[3] eq np and (type[4] eq 0 or EulerPhi(d) le type[4])];

        //printf "Have to check %o possible automorphism-types\n", #PossibleTypes;

        for type in PossibleTypes do
          s := type[4];

          detp := p^s;
          for i := 5 to #type by 3 do
            detp *:= type[i]^type[i+2];
          end for;

          // Enumerate ideal-lattices over K(zeta_m) with given determinant
            K<z> := CyclotomicField(m);
            Kpos := sub<K | z + z^(-1)>;

              A := ClassesModGalois(K);
              M, U, FundUnits := EmbeddingMatrix(K, Kpos);
              LpList := IdealLattices(detp, K, Kpos, A, M, U, FundUnits, false);

              LpList := [L : L in LpList | Minimum(L) ge min];
            LpList := ReduceByIsometry(LpList);

            if s eq 0 then
            Results cat:= [L : L in LpList | Minimum(L) ge min];
            continue m;
            end if;

            for Lp in LpList do
            sigmapList := [c[3] : c in ConjugacyClasses(AutomorphismGroup(Lp)) | MiPoQuotient(c[3], Lp, p) eq Polynomial(GF(p), CyclotomicPolynomial(d))];
              if #sigmapList eq 0 then
```

```
              continue Lp;
              end if;
            "Enumerate candidates for L_1";

            K<z> := CyclotomicField(p);
            Kpos := sub<K|z+1/z>;

              if p eq 2 then

              // In this case use the sublattice U of L_1 with U^{#,2} = U
              det1U := 1;
              for i := 5 to #type by 3 do
                det1U *:= type[i]^type[i+1];
              end for;

              UList := EnumerateGenusDeterminant(det1U, n1, false);

              L1List := &cat[SuperLatticesJuergens(LatticeWithGram(2*GramMatrix(U)), p, Integers() ! ((n1 - s)/2)) : U in UList | Dimension(U) eq 0 or Minimum(U) ge Ceiling(min/2)];
              L1List := [L : L in L1List | Dimension(L) eq 0 or (IsEven(L) and Minimum(L) ge min)];

            elif IsPrime(l) then
              // In this case the genus symbol of L_1 is known and allows for a more efficient enumeration
              k1 := type[6];
              kp := type[7];

              if p le 3 then
                f := 1;
              else
                f := InertiaDegree(Factorization(ideal<Integers(Kpos) | l>)[1][1]);
              end if;
              deltap := (-1)^(Integers() ! (kp/f + (p-1)/2 * (Binomial(Integers() ! (np / (p-1) + 1), 2) + Binomial(s, 2))));
              delta1 := deltap * (-1)^(Integers() ! (s*(p-1)/2));

              if l eq 2 then
                if IsDivisibleBy(np + s*(p-1), 8) then
                  epsilonp := deltap;
                else
                  epsilonp := -deltap;
                end if;

                if IsDivisibleBy(n, 8) then
                  epsilon := 1;
                else
                  epsilon := -1;
                end if;
```

```
        else
          epsilonp := (-1)^(Integers() ! (kp / f + (l-1)/2*Binomial(kp,2)));

          if IsDivisibleBy(n*(l+1), 16) then
            epsilon := 1;
          else
            epsilon := -1;
          end if;
        end if;

        epsilon1 := epsilonp*epsilon;

        Sym1 := [* 2, n1 *];
        if l eq 2 then
          Append(~Sym1, <2, k1, epsilon1, 2, 0>);
          Append(~Sym1, <p, s, delta1>);
        else
          if l lt p then
            Append(~Sym1, <l, k1, epsilon1>);
            Append(~Sym1, <p, s, delta1>);
          else
            Append(~Sym1, <p, s, delta1>);
            Append(~Sym1, <l, k1, epsilon1>);
          end if;
        end if;

        L1List := [L : L in EnumerateGenusSymbol(Sym1) | Dimension(L) eq 0 or (IsEven(L) and Minimum(L) ge min)];

      else

        det1 := p^s;
        for i := 5 to #type by 3 do
          det1 *:= type[i]^type[i+1];
        end for;

        L1List := [L : L in EnumerateGenusDeterminant(det1, n1, true) | Dimension(L) eq 0 or Minimum(L) ge min];

      end if;

      for L1 in L1List do
        sigma1List := [c[3] : c in ConjugacyClasses(AutomorphismGroup(L1)) | MiPoQuotient(c[3], L1, p) eq Polynomial(GF(p), CyclotomicPolynomial(d)) and Degree(MinimalPolynomial(c[3])) le EulerPhi(d)];
        if #sigma1List eq 0 then
          continue L1;
        end if;

        "Constructing superlattices";

        if <l,n> in [] then
          for sigma1 in sigma1List do
            for sigmap in sigmapList do
```

```
                    LList cat:= SuperLatticesMagma(CoordinateLattice(OrthogonalSum(L1,Lp)), p, s, DiagonalJoin(sigma1, sigmap));
                  end for;
                end for;
                elif <l,n> in [<7,18>,<7,20>,<1,24>,<2,24>,<5,24>] then
                LList := [];
                for sigma1 in sigma1List do
                  for sigmap in sigmapList do
                    LList cat:= SuperLattices(CoordinateLattice(L1), CoordinateLattice(Lp), p, sigma1, sigmap);
                  end for;
                end for;
                else
                LList := SuperLatticesJuergens(CoordinateLattice(OrthogonalSum(L1,Lp)),p,s);
                end if;

              Results cat:= [L : L in LList | Minimum(L) ge min];
            end for;
          end for;
        end for;
      end for;
    end for;
  end for;

  return ReduceByIsometry(Results);

end function;


procedure MainLoop()
  for n := 2 to 36 by 2 do
    for l in [1,2,3,5,6,7,11,14,15,23] do
      if l eq 1 and n in [2,4,6] then continue; end if;
      if l eq 2 and n eq 2 then continue; end if;
      if l eq 11 and n in [20,24,28,30,32,34,36] then continue; end if;
      if l eq 23 and n ge 6 then continue; end if;
      printf "dim = %o, l = %o\n", n, l;
      Results := ConstructLattices(l, n);
      ModList := [L : L in Results | IsModular(L, l)];
      StrongModList := [L : L in Results | IsStronglyModular(L,l)];
      PrintFileMagma(Sprintf("SubidealLattices/%o-Modular/%o-Dimensional", l, n), Results : Overwrite := true);
      PrintFileMagma(Sprintf("SubidealLattices/%o-Modular/%o-DimensionalModular", l, n), ModList : Overwrite := true);
      PrintFileMagma(Sprintf("SubidealLattices/%o-Modular/%o-DimensionalStronglyModular", l, n), StrongModList : Overwrite := true);

```

```
      if #Results gt 0 then
        printf "\n\n----------n = %o, l = %o: %o lattices found! %o of them are modular and %o are strongly modular----------\n\n", n, l, #Results, #ModList, #StrongModList;
      end if;
    end for;
  end for;
end procedure;
```

§ 6.5 `MAGMA`-Implementierungen zur Aufzählung der charakteristischen Polynome

Es folgt der Quellcode zu der Methode aus Abschnitt (4.7). Die implementierten Funktionen sind in der Reihenfolge des Quellcodes:

- Einschränken der Möglichkeiten für Automorphismentypen extremaler modularer Gitter, indem versucht wird, Geschlechter von Bild- und Fixgitter mit Dimension ≤ 12 aufzuzählen.

- Liste aller möglichen charakteristischen Polynome, die zu möglichen Automorphismentypen passende Partitionen des Raumes ergeben. Siehe Abschnitt (4.7).

```
load "SubidealLattices.m";

function RestrictAutomorphismTypes(l,n)
// Input: l in N; n in N

// Output: Restricts the possible automorphism types for extremal modular lattices as much as possible

  min := ExtremalMinimum(l,n);

  Types := AutomorphismTypes(l, Integers() ! (n/2), n, min);

  RestrictedTypes := [];

  for type in Types do
    type;
    p := type[1];
    p := type[1];
    n1 := type[2];
    np := type[3];
    s := type[4];

    if p ne 2 and IsPrime(l) then

      k1 := type[6];
      kp := type[7];

      K<z> := CyclotomicField(p);
      Kpos := sub<K | z + z^(-1)>;

      f := InertiaDegree(Factorization(ideal<Integers(Kpos) | l>)[1][1]);
      deltap := (-1)^(Integers() ! (kp/f + (p-1)/2 * (Binomial(Integers() ! (np / (p-1) + 1), 2) + Binomial(s, 2))));
      delta1 := deltap * (-1)^(Integers() ! (s*(p-1)/2));

      if l eq 2 then
        if IsDivisibleBy(np + s*(p-1), 8) then
          epsilonp := deltap;
        else
          epsilonp := -deltap;
        end if;

        if IsDivisibleBy(n, 8) then
          epsilon := 1;
        else
          epsilon := -1;
        end if;
      else
        epsilonp := (-1)^(Integers() ! (kp / f + (l-1)/2*Binomial(kp,2)));

        if IsDivisibleBy(n*(l+1), 16) then
          epsilon := 1;
        else
          epsilon := -1;
```

```
    end if;
   end if;

   epsilon1 := epsilonp*epsilon;

   Sym1 := [* 2, n1 *];
   Symp := [* 2, np *];
   if l eq 2 then
    Append(~Sym1, <2, k1, epsilon1, 2, 0>);
    Append(~Sym1, <p, s, delta1>);
    Append(~Symp, <2, kp, epsilonp, 2, 0>);
    Append(~Symp, <p, s, deltap>);
   else
    if l lt p then
     Append(~Sym1, <l, k1, epsilon1>);
     Append(~Sym1, <p, s, delta1>);
     Append(~Symp, <l, kp, epsilonp>);
     Append(~Symp, <p, s, deltap>);
    else
     Append(~Sym1, <p, s, delta1>);
     Append(~Sym1, <l, k1, epsilon1>);
     Append(~Symp, <l, kp, epsilonp>);
     Append(~Symp, <p, s, deltap>);
    end if;
   end if;

   if n1 le 12 and n1 gt 0 then
    List := [L : L in EnumerateGenusSymbol(Sym1) | Minimum(L) ge min];
    if #List eq 0 then
     continue type;
    end if;
   end if;

   if np le 12 and np gt 0 then
    List := [L : L in EnumerateGenusSymbol(Symp) | Minimum(L) ge min];
    if #List eq 0 then
     continue type;
    end if;
   end if;

  else

   if n1 le 12 and n1 gt 0 then
    det1 := p^s;
    for i := 5 to #type by 3 do
     det1 *:= type[i]^type[i+1];
    end for;

    List := [L : L in EnumerateGenusDeterminant(det1, n1, true) | Minimum(L) ge min];
    if #List eq 0 then
     continue type;
    end if;
   end if;
```

```
      if np le 12 and np gt 0 then
        detp := p^s;
        for i := 5 to #type by 3 do
          detp *:= type[i]^type[i+2];
        end for;

        List := [L : L in EnumerateGenusDeterminant(detp, np, true) | Minimum(L) ge min];
        if #List eq 0 then
          continue type;
        end if;
      end if;

    end if;

    Append(~RestrictedTypes, type);
  end for;

  return RestrictedTypes;

end function;


function PossibleCharPos(l, n)
// Input: l in N; n in N

// Output: List of all characteristic polynomials of lattices possibly not found by the subideal-lattice algorithm. Format: [[<d_1,c_1>,...,<d_k,c_k>], ...] for the exponents c_i > 0 of the Phi_(d_l) for the divisors d_l

  Types := RestrictAutomorphismTypes(l,n);

  Results := [];

  for phim in [Integers() ! (n/2)+1..n] do
    for m in EulerPhiInverse(phim) do
      Div := Sort(Divisors(m));
      Phi := [EulerPhi(d) : d in Div];
      k := #Div;

      pList := [p : p in PrimeDivisors(m) | Gcd(p,l) eq 1];
      FixDimLists := [];
      for p in pList do
        FixDims := [];
        for type in Types do
          if type[1] eq p then
            FixDim := type[2];
            if not FixDim in FixDims then
              Append(~FixDims, FixDim);
            end if;
          end if;
        end for;
        if #FixDims eq 0 then
          continue m;
```

```
      end if;
      Append(~FixDimLists, FixDims);
    end for;

    t := #pList;

    M := ZeroMatrix(Integers(), k, t+1);
    for i in [1..k] do
      for j in [1..t] do
        if IsDivisibleBy(Integers() ! (m/pList[j]), Div[i]) then
          M[i,j] := Phi[i];
        end if;
      end for;
      M[i, t+1] := Phi[i];
    end for;

    if t gt 0 then
      TypeChoice := CartesianProduct([[1..#List]: List in FixDimLists]);

      for IndexList in TypeChoice do
        N := ZeroMatrix(Integers(), 1, t+1);
        MaxDim := [];
        for i in [1..t] do
          N[1][i] := FixDimLists[i][IndexList[i]];
        end for;
        N[1][t+1] := n;

        MaxDim := [Floor(n/d) : d in Div];
        for i in [1..k] do
          for j in [1..t] do
            if IsDivisibleBy(Integers() ! (m / pList[j]), Div[i]) then
              MaxDim[i] := Minimum(MaxDim[i], Floor(N[1][j] / Phi[i]));
            else
              MaxDim[i] := Minimum(MaxDim[i], Floor((n-N[1][j]) / Phi[i]));
            end if;
          end for;
        end for;

        C := CartesianProduct([[0..MaxDim[i]] : i in [1..k]]);

        for c in C do
          v := Matrix(Integers(), 1, k, [x : x in c]);
          if v*M eq N then
            if Lcm([Div[i] : i in [1..k-1] | c[i] gt 0]) eq m then
              ExpList := [<Div[i], c[i]> : i in [1..k] | c[i] gt 0];
              Append(~Results, ExpList);
            end if;
          end if;
```

```
          end for;
        end for;
      else
        C := CartesianProduct([[0..Floor(n/EulerPhi(d))] : d in Div]);
        N := Matrix(Integers(), 1, 1, [n]);
        for c in C do
          v := Matrix(Integers(), 1, k, [x : x in c]);
          if v*M eq N then
            if Lcm([Div[i] : i in [1..k] | c[i] gt 0]) eq m then
              ExpList := [<Div[i], c[i]> : i in [1..k] | c[i] gt 0];
              Append(~Results, ExpList);
            end if;
          end if;
        end for;
      end if;
    end for;
  end for;

  return Results;

end function;
```

§ 6.6 Eidesstattliche Versicherung

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel *Extremale Gitter mit großen Automorphismen* selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, im September 2018 ____________________________

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe von bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe von bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, im September 2018 ____________________________

7 Literaturverzeichnis

[BFS05] Eva Bayer Fluckiger and Ivan Suarez. Modular lattices over cyclotomic fields. *Journal of Number Theory*, 114:394–411, 2005.

[CE03] Henry Cohn and Noam Elkies. New upper bounds on sphere packings I. *Annals of Mathematics*, 157:689–714, 2003.

[CS93] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*, volume 290 of *Grundlehren der mathematischen Wissenschaften*. Springer, 3rd edition, 1993.

[Jü15] Michael Jürgens. *Nicht-Existenz und Konstruktion extremaler Gitter*. PhD thesis, Technische Universität Dortmund, März 2015.

[Kne02] M. Kneser. *Quadratische Formen*. Springer, 2002.

[Lor11] David Lorch. Einklassige Geschlechter positiv definiter dreidimensionaler Gitter, 2011.

[Mol11] Richard A. Mollin. *Algebraic number theory*. CRC Press, 2nd edition, 2011.

[Neb13] Gabriele Nebe. On automorphisms of extremal even unimodular lattices. *International Journal of Number Theory*, 09:1933–1959, 2013.

[Neu92] Jürgen Neukirch. *Algebraische Zahlentheorie*. Springer, 1992.

[NS] Gabriele Nebe and N. J. A. Sloane. A Catalogue of Lattices. http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/. Aufgerufen: 10.08.2018.

[Que95] H. G. Quebbemann. Modular Lattices in Euclidean Spaces. *Journal of Number Theory*, 54:190–202, 1995.

[SH98] Rudolf Scharlau and Boris Hemkemeier. Classification of integral lattices with large class number. *Mathematics of computation*, 67(222):737–749, April 1998.