Rigorously Analyzed Algorithms for the Discrete Logarithm Problem in Quadratic Number Fields

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erreichung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Math. Ulrich Vollmer

aus Berlin

Referenten:	Prof. Dr. Johannes Buchmann (TU Darmstadt)
	Prof.dr. Peter Stevenhagen (Universiteit Leiden, NL)
Tag der Einreichung:	15. September 2003
Tag der mündlichen Prüfung:	28. Oktober 2003

Darmstadt, 2003
Hochschulkennziffer: D 17

To my parents

and

To all who were denied an academic education
by political arbitrariness

Acknowledgements

At this point I would like to take the opportunity to thank at least some of those people whose support enabled me to do the research which led to this thesis.

First of all I would like to thank my thesis advisor, Prof. Dr. Johannes Buchmann, for proposing the topic, for his advice and support and the investment of time for discussions that provided ideas and impetus especially at the important initial phase of my thesis research. He has proved to be a wonderful guide into the world of academia.

I would like to thank Prof.dr. Stevenhagen for evaluating and carefully reading this thesis on short notice, and the detailed comments for improvement he made.

I would like to thank the Deutsche Forschungsgemeinschaft for financing the project on Number Field Cryptography and thereby providing the financial support for our Number Field Cryptography group comprised of Safuat Hamdy, Andreas Meyer and myself.

I would like to thank those mathematicians who introduced me to the richness and beauty of mathematics in general and number theory in particular, among which I would like to mention in particular Prof. Dr. Klaus Haberland (FSU Jena), and Prof. Glenn Stevens, Ph.D. (BU, Boston). I would also like to thank the EPN, Quito, and especially Prof. Hallo for providing me with the opportunity to develop and teach my first cryptography course. The interest of professors and students of this institution has done much to motivate me to cross from mathematics into computer science.

I would like to thank my colleagues at the Theoretical Computer Science group in Darmstadt for the friendly, stimulating atmosphere they contributed so much to, and in particular my long-time office-mate Safuat Hamdy.

Last, but far from least I would like to thank my wife, Susan Stewart, for her never ceasing love, tolerance and support during the turbulent years during which we both completed our PhD while raising our wonderful daughters Anita and Sylvia; and my parents who longed so much to see this work completed.

Zusammenfassung

Das Ziel der vorliegenden Arbeit besteht in der Analyse der Komplexität des Diskreten-Logarithmus-Problems (DLP) und verwandter Probleme in quadratischen Ordnungen.

Die Bedeutung dieser Probleme entspringt in erster Linie ihrer Anwendung für die Quadratische-Zahlkörper-Kryptographie (QNFC). QNFC wurde von Buchmann und Williams [BW88], [BW90] vorgeschlagen und beruht auf der Annahme, daß das DLP in quadratischen Ordnungen schwer zu lösen ist. Wenn wir die Frage beantworten wollen, ob und ab welcher Schlüsselgröße QNFC sicher ist, müssen wir die Komplexität des DLP studieren.

Diese Promotionsarbeit verfolgt einen rigorosen theoretischen Ansatz. Im Falle der probabilistischen Algorithmen legt die Analyse die wohlfundierte und weithin akzeptierte Verallgemeinerte Riemannsche Hypothese zugrunde.

Ausgangspunkt für die vorgestellten deterministischen Algorithmen waren die Arbeiten von Shanks [Sha71], [Sha72] und die Abwandlung der Shanksschen Idee für allgemeine Gruppen durch Terr [Ter00]. Die Analyse der Laufzeit ist detailliert und weist neben der dominierenden Wurzel des Regulators die geringstmögliche Potenz aus, mit welcher der Logarithmus der Diskriminante in die Laufzeitschranke eingeht.

Ausgangspunkt für die vorgestellten probabilistischen Algorithmen waren die Arbeit von Hafner und McCurley [HM89] für negative Diskriminanten und die Arbeit von Buchmann [Buc90] für positive. Der Kern des verfolgten Ansatzes liegt in der Beschränkung der Berechnungen auf die essentiellen Daten (insbesondere partielle statt voller Relationengitter und beweisbar dünn besetzte Basen) und dem Einsatz schnellerer Teilalgorithmen.

Ein solcher Teilagorithmus ist ein neu entwickelter Algorithmus für die Berechnung der Hermite-Normalform (HNF) einer ganzzahligen Matrix in kubischer Zeit.

In der Konsequenz erhalten wir die derzeit schnellsten deterministischen und probabilistischen Algorithmen für die untersuchten Probleme und eine obere Schranke für die Komplexität des DLP.

Abstract

The purpose of this thesis is to study the complexity of the discrete logarithm problem (DLP) and related problems in quadratic orders.

The significance of these problems stems in large part from their application in quadratic number field cryptography (QNFC). QNFC was proposed by Buchmann and Williams [BW88], [BW90], and relies on the fact that the DLP in quadratic orders is hard. The question whether QNFC is secure at all, and the choice of cryptographically secure key-sizes, if it is, requires the study of the difficulty of the DLP.

This thesis takes a rigorous theoretical approach to this question. In the case of the probabilistic algorithms the analysis relies on the well-established Generalized Riemann Hypothesis (GRH).

Starting point for the proposed deterministic algorithms were the works of Shanks [Sha71], [Sha72] and the variant of Shanks' idea for general groups suggested by Terr [Ter00]. We give a detailed analysis of the run-time of our algorithms and indicate apart from the square root of the regulator at which (minimzed) power the logarithm of the discriminant enters the run-time bound.

Starting point for the proposed probabilistic algorithms were the work of Hafner and McCurley [HM89] for negative discriminants and of Buchmann [Buc90] for positive ones. The heart of the presented approach lies in the restriction of the computations to essential data (in particular partial instead of full relation lattice and sparse bases) and the use of fast sub-algorithms.

One of these sub-algorithms is a newly developed algorithm for the computation of the Hermite Normal Form (HNF) of an integer matrix in cubic time.

In summary we obtain the currently fastest deterministic and probabilistic algorithms for the problems at hand, and gives a rigorous analysis of their properties. Thus we establish an upper bound for the complexity of the DLP.

Inhaltsverzeichnis

List of Algorithms

Kapitel 1

Introduction

The purpose of this thesis is to study the complexity of the discrete logarithm problem (DLP) and related problems in quadratic orders. The significance of these problems stems in large part from their application in quadratic number field cryptography (QNFC). QNFC relies on the fact that the DLP in quadratic orders is hard. The question whether QNFC is secure at all, and the choice of cryptographically secure key-sizes, if it is, requires the study of the difficulty of the DLP.

This thesis takes a rigorous theoretical approach to this question. It presents the currently fastest deterministic and probabilistic algorithms for the problems at hand, and gives a rigorous analysis of their properties. In the case of the probabilistic algorithms the analysis relies on the well-established Generalized Riemann Hypothesis (GRH).

Thus we establish an upper bound for the complexity of the DLP. For concrete cryptographic applications the results of this thesis need to be combined with experiments with state-of-the-art practical algorithms, see, e.g., [Jac00], which unfortunately at the current time lack rigorous analysis.

1.1 The problems

Let G be a commutative group. In this thesis we will take G to be the group of fractional ideals or the class group of a quadratic order. Henceforth we will write G multiplicatively, and let 1 denote its neutral element. In this context, we will analyze the algorithmic complexity of the following problems.

Problem 1 (Element Order Problem)
Given $g \in G$, determine its order, i.e. minimal natural $n \in \mathbb{N}$ such that $g^n = 1$.

Problem 2 (Discrete Logarithm Problem)
Given $g, h \in G$, determine the discrete logarithm of h with basis g. More precisely, given $g, h \in G$ determine whether h is in the subgroup of G

generated by g, and, if this the case, minimal $n \in \mathbb{N}$ such that $g^n = h$.

We will sometimes drop the minimality condition and speak of the "relaxed discrete logarithm problem."

Problem 3 (Extended Discrete Logarithm Problem)
Given a group P mapped homomorphically to G by $\phi : P \longrightarrow G$, and $g, h \in G$, determine the discrete logarithm of h with respect to g relative to P.

More precisely, given $\phi : P \longrightarrow G$, and $g, h \in G$ determine whether h lies in the subgroup of G generated by g and $\phi(P)$, and, if this is the case, do the following.

- Find minimal $n \in \mathbb{N}$ such that there exists an $a \in P$ with $\phi(a) \cdot g^n = h$. This n will be called the *exponent* of the discrete logarithm relation.
- Determine some $a \in \phi^{-1}(g^{-n}h)$. Such a will be called a *generator* of the discrete logarithm relation.

In our situation we will take P to be the multiplicative group of the fraction field of the order in question. Then ϕ maps a number to the principal ideal it generates.

The final set of problems concerns natural questions about the group G itself.

Problem 4 (Group Order Problem)
Determine the cardinality of G.

Problem 5 (Group Structure Problem)
Determine the invariants of G. More precisely, find elements $h_1, \ldots, h_l$ in G and natural numbers $s_1, \ldots, s_l$ such that $\operatorname{ord} h_i = s_i$, $s_{i+1} \mid s_i$, and

$$G = \langle h_1 \rangle \oplus \cdots \oplus \langle h_l \rangle$$

The last problem can be reduced to the following:

Given a sequence $\mathcal{G} = (g_1, \ldots, g_n)$ such that $\{ g_i \mid i = 1, \ldots, n \}$ generates G find relations $v_1, \ldots, v_n \in \mathbb{Z}^n$ such that $\{ v_j \mid j = 1, \ldots, n \}$ generates the lattice of relations on $\mathcal{G}$, and the matrix H that has v_i as column vectors is in Hermite normal form.

The computation of the invariants is then just a matter of computing the Smith Normal Form (SNF) S of H, and transforming matrices U and V such that $S = U * H * V$. For SNF algorithms see, e.g., [Sto98].

The time required for the reduction is cubic in n. In the case of class groups of quadratic orders n can be chosen quadratic in the logarithm of the discriminant, pre-supposing the validity of the GRH.

1.2 Application in Number Field Cryptography

In this section we give a brief overview over public-key crypto-systems whose security depends on the difficulty of solving the problems listed in the previous section. For a more comprehensive overview, and a detailed list of references see [BTV04].

The Diffie-Hellman key exchange protocol (DH-KE) and the Diffie-Hellman integrated encryption scheme (DH-IES) depend on the difficulty of the Diffie-Hellman (DH) problem in a group G.

Problem 6 (Diffie-Hellman problem)
Given $g \in G$ and two group elements g^a and g^b in $\langle g \rangle$, find g^{ab}.

The class group Cl_Δ of an imaginary quadratic order is a suitable candidate for the group G. The resulting schemes are called IQ-DH and IQ-IES.

The Diffie-Hellman problem can obviously be reduced to the Discrete-Logarithm Problem. Henceforth, the algorithms given in this thesis yield upper bounds for the difficulty of the DH Problem in $G = \mathrm{Cl}_\Delta$.

The DSA signature scheme is based on the difficulty of the DLP in the used group G. It requires, however, the knowledge of the group order. There is a variant of DSA which works in the class group of an imaginary quadratic order, called IQ-DSA. The security of IQ-DSA hinges likewise on the difficulty of the DLP in Cl_Δ.

The Guillou-Quisqater (GQ) signature scheme relies on the difficulty of the root problem (RP) in the used group.

Problem 7 (Root problem)
Given $g \in G$ and $n \in \mathbb{N}$ find $h \in G$ such that $h^n = g$.

The GQ scheme does not require the knowledge of the group order. It can thus be implemented in the class group of an imaginary quadratic order. The resulting scheme will be abbreviated IQ-GQ. Note, however, that the RP is slightly weaker than the DLP.

In 1989 Buchmann and Williams introduced a variant of the Diffie-Hellman key exchange protocol which works in the infrastructure of a real quadratic order, called RQ-DH. Its security depends on the difficulty of the Principal Ideal Problem (PIP). The PIP is a special case of the EDLP. It requires to find a generator for a given principal ideal.

Building on RQ-DH, it is possible to implement ElGamal encryption. There is a signature scheme whose security relies on the difficulty of the PIP, too. This is a variant of the Fiat-Shamir signature protocol, and, hence, called PIP-FS.

The asymptotic run-time bounds we will prove in this thesis—based on a Generalized Riemann Hypothesis—for the probabilistic algorithms proposed herein are smaller than any other such bounds which have been proved

before. Combined with performance data from practical computations, they yield lower bounds for secure cryptographic key-sizes for IQ-DH, IQ-IES, IQ-DSA, IQ-GQ, RQ-DH and PIP-FS. For an outline of how this can be done, see Hamdy's works [HM00] und [Ham02]. In his analysis, however, Hamdy uses a heuristic run-time bound for IQ-DL algorithms which is—from the point of view of security—more cautious than the one proven here.

1.3 Prior work

Deterministic algorithms. The baby-step giant-step idea goes back to Shanks, see [Sha71]. He gave generic algorithms for computing the element order, solving the DLP and GSP. The algorithms assume the knowledge of an upper bound B for the group order. Their complexity (in terms of group operations) is in $O(\sqrt{B})$. An analysis of a variant of the generic baby-step giant-step algorithm that does not require this knowledge was given by Buchmann, Jacobson and Teske in [BJT97].

The deterministic algorithms in this work start from another variant of the baby-step giant-step algorithm given by Terr for the EOP in [Ter00]. The complexity of this algorithm is $O(\sqrt{\operatorname{ord} g})$.

Probabilistic algorithms The index calculus approach goes back to Kraitchik [Kra22], and later but seemingly independent, to [WM68]. See [Odl00] for a brief account of the developments in the years since.

The index calculus approach was first systematically applied to discrete logarithms in finite fields with prime order by Pollard, Adleman and Merkle. It was later carried over to arbitrary finite fields, and applied to the class group of imaginary quadratic orders for factorization purposes.

Seysen [Sey87] proposed the first algorithm of the latter type for which run-time bounds were provable. Apart from relying on an Extended Riemann Hypothesis, the proof he gave was rigorous.

Hafner and McCurley [HM89] built on Seysen's ideas to propose an index calculus algorithm for the GSP in the class group of a maximal imaginary quadratic order. Their algorithm was later generalized by Buchmann [Buc90] to the GOP and Regulator problem in arbitrary global number fields. For real quadratic fields, his analysis relies solely on a Generalized Riemann Hypothesis.

Finally, Buchmann and Düllmann published a series of papers, see e.g. [BD91a] and [BD91b] adapting Hafner/McCurley's and Buchmann's algorithms to the DLP and the EDLP.

It should be noted that the analysis of all the algorithms mentioned so far relies on a Riemann hypothesis. It is therefore interesting to ask whether one can abandon this premise.

We mention two examples: Lenstra and Pomerance gave in [LP92] an algorithm for factoring integers whose analysis does not depend on a Riemann

Hypothesis. Lovorn and Pomerance presented in [BP98] a rigorous analysis for the DLP in finite fields.

There are currently no algorithms known for any of the stated problems in class groups of quadratic orders whose run-time can be proved without the premise of a Riemann hypothesis.

Run times. Run time bounds for probabilistic algorithms in this thesis will be given in terms of the function

$$L_x(a,b) = \exp(b(\log x)^a(\log\log x)^{1-a}).$$

For a in $(0,1)$ this function is sub-exponential in the sense that it grows faster than any polynomial in $\log x$, but slower than x^e for any $e > 0$.

Hafner and McCurley's algorithm has run-time $L_{|\Delta|}(\frac{1}{2}, \sqrt{2}+o(1))$, where Δ is the discriminant of the order in question.

Buchmann gave a run-time bound of $L_D(\frac{1}{2}, 9/(2\sqrt{7}) + o(1))$ for his algorithm where D is the absolute discriminant of the field in question. This was later improved by Abel in her thesis [Abe94] to $L_\Delta(\frac{1}{2}, 5/6\sqrt{3} + o(1))$.

1.4 Results

Deterministic algorithms. In this thesis we give two algorithms: one for the computation of the regulator, and one for the solution of the EDLP in a real quadratic order. Both employ Terr's novel approach to sequencing the steps in a baby-step giant-step algorithm. The use of this approach enables us to present algorithms which are considerably simpler than their pre-cursors given in [BJT97].

The first algorithm has complexity $O((\log\Delta + \sqrt{R_\Delta}) \cdot (\log\Delta)^2)$. Here Δ is the discriminant of the order, R_Δ is its regulator, and $\mu(s)$ denotes the time required for multiplying two s-bit numbers. The second algorithm has complexity $O(\sqrt{n}(\log\Delta + \sqrt{2R\log 2}) \cdot (\log\Delta)^2)$, where n is the sought exponent of the discrete logarithm, and the other parameters are as before.

The exponent with which the logarithm of the discriminant enters into the run-time bounds appears to be lower than in competing algorithms where the analysis has not been performed in equal detail.

For a discussion of the mean-time advantages of the Terr variant which also apply to the algorithms here, see [Ter00].

Probabilistic algorithms. We give algorithms for the EOP, the GOP, the GSP, and the (E)DLP in maximal quadratic orders. The algorithms are rigorously analyzed on the basis of a Generalized Riemann Hypothesis (GRH). Their run-time is bounded by $L_{|\Delta|}(\frac{1}{2}, 3/\sqrt{8} + o(1))$. Preliminary versions of the results presented here have appeared in [Vol00] and [Vol02].

The novel ideas which enable this improvement over prior results are:

- The reduction to the computation of relation lattices which are only "essentially full", i.e. full if restricted to the rows corresponding to a sub-set of the factor base that generates the class group.
- The generation of only sparse relations combined with the proof that they suffice to generate essentially full relation lattices.
- ($\Delta > 0$) Substitution of the computation of the full kernel of the integral part of the relation lattice by the computation of just two dependencies combined with the proof that these two dependencies very likely yield the full unit group.
- The use of Bernstein's factoring algorithm for the detection of smooth numbers in average polynomial time.
- A cubic algorithm for the computation of determinant and the essential part of the HNF of relation matrices based on the solution of large Diophantine systems.

The run-time asymptotics of the proposed algorithms matches the observed run-time behavior of the state-of-the-art heuristically analyzed algorithms, e.g., that of Jacobson's IQ-MPQS, see [Jac99].

1.5 Complexity

All functions used to bound the complexity of the algorithms given are functions of the main parameter Δ, the discriminant of the order in which the computations are performed.

For two functions f, g we write $f = O(g)$ if there exists $c \in \mathbb{R}_{>0}$ such that $f(\Delta) < c \cdot g(\Delta)$ for $|\Delta| \gg 0$. We write $f = o(g)$ if for all $c \in \mathbb{R}_{>0}$ we have $f(\Delta) < c \cdot g(\Delta)$ whenever $|\Delta| \gg 0$.

In Chapter 4 we will also use "Soft-Oh"-notation: $f = O^{\sim}(g)$ if there exist $c \in \mathbb{R}_{>0}$ and $k \in \mathbb{N}$ such that $f(\Delta) < g(\Delta) \cdot \log^k(|\Delta|)$ for $|\Delta| \gg 0$.

Note the following obvious relations between complexity classes involving the function $L_{|\Delta|}(a, b)$. We have

$$L_{|\Delta|}(a, b + o(1)) \cdot L_{|\Delta|}(a, b' + o(1)) = L_{|\Delta|}(a, b + b' + o(1))$$
$$L_{|\Delta|}(a, b + o(1)) + L_{|\Delta|}(a, b' + o(1)) = L_{|\Delta|}(a, \max(b, b') + o(1))$$

If $f = O^{\sim}(1)$, then

$$f \cdot L_{|\Delta|}(a, b) = L_{|\Delta|}(a, b + o(1)).$$

All constants hidden in the introduced notations will be explicitly computable, unless explicitly stated otherwise.

Kapitel 2

Reduction

In this chapter we provide the background for the analysis of the algorithms presented in the following chapters. We list standard results on reduction of ideals in quadratic orders.

Good references for the material of this chapter are [Len82], and [BTW95]. Unless otherwise noted, statements given here without proof follow directly from the results presented in these articles.

2.1 Ideals

Throughout this thesis $\mathcal{O}$ denotes a quadratic order, Δ its discriminant, and $\mathcal{K}$ its quotient field. We fix an embedding of $\mathcal{K}$ into $\mathbb{C}$ or $\mathbb{R}$, depending on the sign of Δ, and denote conjugation by σ. The norm of $\alpha \in \mathcal{K}$ is $\mathrm{N}(\alpha) = \alpha\alpha^\sigma$. The order $\mathcal{O}$ has integral basis $(1, (\Delta + \sqrt{\Delta})/2)$ over $\mathbb{Z}$.

A non-zero $\mathcal{O}$-sub-module $\mathfrak{a}$ of $\mathcal{K}$ is called a *fractional $\mathcal{O}$-ideal* if and only if there exists a $d \in \mathbb{Z}$ such that $d\mathfrak{a} \subset \mathcal{O}$. It is called invertible if there exists a fractional ideal $\mathfrak{b}$ such that $\mathfrak{a}\mathfrak{b} = \mathcal{O}$. The group of invertible ideals will be denoted by I_Δ.

Fractional ideals will be called simply "ideals" in all that follows. If we want to emphasize that an ideal is contained in $\mathcal{O}$, then we will call it integral.

Principal ideals $\alpha \cdot \mathcal{O}$ with $\alpha \neq 0$ are always invertible. They form a subgroup of I_Δ that is denoted by P_Δ. The factor group I_Δ / P_Δ is the *class group* of $\mathcal{O}$ (in the ordinary sense). The class (1) mod P_Δ is called the principal class. A class whose square equals the principal class is called ambiguous.

The unit group of $\mathcal{O}$ will be denoted by $\mathcal{O}^*$. If $\Delta < 0$, then it is finite. If $\Delta > 0$, then it is the product of (± 1) with $\mathcal{O}^*_+$, the subgroup of positive units which is cyclic. There is exactly one generator of $\mathcal{O}^*_+$ which is greater than 1. This is called the *fundamental unit* of $\mathcal{O}$, and denoted by ε_Δ. Its logarithm is the *regulator* of $\mathcal{O}$, and denoted by R_Δ.

For an integral ideal $\mathfrak{a}$ we will call the minimum of $\mathfrak{a} \cap \mathbb{N}$ its *exponent*,

and denote it by $\exp(\mathfrak{a})$. For a non-integral ideal $\mathfrak{a}$ we will call minimal $d \in \mathbb{N}$ such that $d\mathfrak{a}$ is integral its *integral denominator*. Since we will make no use of the more frequently used concept of denominator of a fractional integral $\mathfrak{a}$ which denotes the integral ideal $\mathfrak{c}$ with smallest norm such that $\mathfrak{c}\cdot\mathfrak{a}$ is integral, we will drop the attribute "integral" in "integral denominator" and speak simply of *denominator*.

We will use a *standard representation* for fractional ideals in $\mathcal{O}$. Given $\mathfrak{a} \in I_\Delta$, there are d, a, and b, all in $\mathbb{Z}$, such that

$$\mathfrak{a} = \frac{1}{d} \cdot \left(a\mathbb{Z} + \frac{b+\sqrt{\Delta}}{2}\mathbb{Z} \right), \tag{2.1}$$

where $a, d > 0$ and $\gcd(a, b, (b^2 - \Delta)/(4a)) = 1$. The choice of a and d is unique. The exponent of $\mathfrak{a}$ eaquals a. The denominator of $\mathfrak{a}$ equals d. The choice of b is unique mod $2a$. We will choose b to come from the interval $(-a, a]$ if $\Delta < 0$ or $\Delta > 0$ and $a > \sqrt{\Delta}$. Otherwise, we will choose the interval $(\sqrt{\Delta} - 2a, \sqrt{\Delta})$.

An integral ideal $\mathfrak{a}$ is called *primitive* if there is no $d \in \mathbb{Z}$ such that $d \neq \pm 1$ and $(1/d) \cdot \mathfrak{a}$ is integral, too. For primitive ideals the notions of exponent and norm coincide.

A primitive ideal is called *reduced* if there is no non-zero $\alpha \in \mathfrak{a}$ such that $|\alpha|$ and $|\alpha^\sigma|$ are both smaller than $\exp(\mathfrak{a})$. In terms of the standard representation (2.1) this means that

$$|\sqrt{\Delta} - a| < |b| < \sqrt{\Delta}.$$

Note the deviation from the classical definition of reducedness which omits taking the absolute value of b. The definition given here has the advantage that an ideal is reduced if and only if its conjugate is. Moreover, it generalizes to number fields of arbitrary degree.

The set of reduced ideals will be denoted by $\mathcal{R}_\Delta$. If $\Delta < 0$, then there is exactly one in each non-ambiguous class and exactly two conjugated reduced ideals in each ambiguous ideal class. If $\Delta > 0$, then the set of all reduced ideals in a given class will be called the *cycle* in this class. It exhibits a certain group-like structure, which is called the infra-structure of $\mathcal{O}$.

2.2 Representation of field elements

The complexity of operations involving elements of $\mathcal{K}$ depends on the "size" of their representation. For a simple measure of the size of a number, we introduce the notion of height. For any $\alpha \in \mathcal{K}$, we define the (naive) height $H(\alpha)$ to be the maximum of $|\alpha|$ and $|\alpha^\sigma|$. The denominator of $\alpha \in \mathcal{K}$ is minimal $d \in \mathbb{N}$ such that $d\alpha \in \mathcal{O}$. It will be denoted by $d(\alpha)$.

The algorithms in this thesis will work with numbers α in $\mathcal{K}$ in one out of possible representations:

1. Standard representation: α is given by the triple $x, y, d \in \mathbb{Z}$ such that $\alpha = \frac{x+y\sqrt{\Delta}}{2d}$ where $\gcd(x, y, d) = 1$ and $d > 0$. This representation will be used for quadratic integers of small height and small denominator. In this case the size of x, y, and d (defined as the length of their binary representation) will be polynomially bounded in $\log \Delta$.

2. Compact representation (see [BTW95]): α is given by

$$\alpha = \gamma \prod_{j=1}^{k} \gamma_j^{2^{k-j}}.$$

   In this work we will have, $k = O(\log \Delta)$, each of the components γ_j is given in standard representation and has small height, and the size of γ is bounded by $O(\log \mathrm{N}(\alpha))$. This representation will be used for numbers with small norm, in particular generators of reduced ideals. We will sometimes write this representation as the vector $(\gamma, \gamma_1, \ldots, \gamma_k)$.

3. Power product representation: α is given by a vector $\big((\gamma, a_\gamma)\big)$ such that

$$\alpha = \prod \gamma^{a_\gamma}$$

   where each γ is in turn given in standard or compact representation and has small norm. The binary length of the exponents a_γ as well as the number of factors will be bounded by a sub-exponential function in $\log \Delta$.

   This is an auxiliary representation. Note that it is possible to convert this representation in polynomial time into compact representation *provided α has small norm.*

Note that for numbers given in compact representation with $O(\log \Delta)$ components and component height H such that $\log H = O((\log \Delta)^c)$ for some fix c we can perform the usual arithmetic operations in polynomial time.

Lemma 2.1 *Given $\alpha, \beta \in K$ in compact representation, and $\mathfrak{a} \in I_\Delta$ in standard representation, it is possible to compute a compact representation of $\alpha \cdot \beta$, and the standard representation of $\alpha\mathfrak{a}$ in time polynomial in the size of α, β, and $\mathfrak{a}$.* ∎

A proof of the assertions of this lemma can be found in the work [BTW95] where compact representations are studied in detail.

2.3 Reduction

There is a reduction map ρ on I_Δ with the following properties: Applying ρ repeatedly to any primitive $\mathfrak{a} \in I_\Delta$ will eventually yield a reduced ideal. The map ρ permutes the set of reduced ideals in a given class. It is defined as follows.

Let $\mathfrak{a}$ be a primitive ideal given in standard representation (2.1). Define the reducing number of $\mathfrak{a}$ by

$$\gamma(\mathfrak{a}) = \frac{b + \sqrt{\Delta}}{2ct}, \tag{2.2}$$

where $c = (b^2 - \Delta)/(4a)$, and $t = \pm 1$ is chosen such that $\gamma(\mathfrak{a}) > 0$. Note that $t = -1$ if $\mathfrak{a}$ is reduced. Let

$$\rho(\mathfrak{a}) = \mathfrak{a}/\gamma(\mathfrak{a}). \tag{2.3}$$

If $\mathfrak{a}$ is reduced, then so is $\rho(\mathfrak{a})$.

Lemma 2.2 *For any ideal $\mathfrak{a} \in \mathcal{R}_\Delta$, computing the standard representation of $\rho(\mathfrak{a})$ has time complexity bounded by $O(\mu(\log \Delta))$.*

Proof This follows from the explicit formulae. ∎

Given $\mathfrak{a} \in I_\Delta$ we will denote the first reduced element of the reduction sequence starting at $\mathfrak{a}$ by $\rho_0(\mathfrak{a})$.

In [BB99], Johannes Buchmann and Ingrid Biehl proved the following two propositions limiting the complexity of the computation of ρ_0.

Proposition 2.3 *Let $\mathfrak{a} \in I_\Delta$ be an invertible $\mathcal{O}$-ideal with exponent a, and $(\mathfrak{a}_d)$ the reduction sequence starting at $\mathfrak{a}$. If $\mathfrak{a}_e$ is the first reduced ideal in this sequence then $e < \log(a/\sqrt{\Delta}) + 2$.*

Proposition 2.4 *There is an algorithm, later denoted* Rho0*, which given as input some integral $\mathfrak{a} \in I_\Delta$ with exponent a computes the reduction sequence $(\mathfrak{a}_d)$ starting at $\mathfrak{a}$ up to and including $\mathfrak{a}_e = \rho_0(\mathfrak{a})$, and the relative generator $\alpha_e = \prod_{i=0}^{e-1} 1/\gamma(\mathfrak{a}_i)$ in time and space bounded from above by $O(\log^2(a/\sqrt{\Delta}))$.*

There is a variation of Rho0 described and proven to run in pseudo-linear time by Arnold Schönhage in [Sch71] . It yields some reduced ideal $\mathfrak{b}$ in the class of a given $\mathfrak{a} \in I_\Delta$ but not a generator of $\mathfrak{b}$ relative to $\mathfrak{a}$ in time $O(\log^{1+\epsilon}(a/\sqrt{\Delta}))$.

We define the operations

$$(\Delta < 0) \quad * : \mathcal{R}_\Delta^2 \longrightarrow \mathcal{R}_\Delta \; : \; (\mathfrak{a}, \mathfrak{b}) \longmapsto \rho_0(\mathfrak{a}\mathfrak{b}) \tag{2.4}$$

$$(\Delta > 0) \quad * : \mathcal{R}_\Delta^2 \longrightarrow \mathcal{R}_\Delta \; : \; (\mathfrak{a}, \mathfrak{b}) \longmapsto (\mathfrak{c}, \gamma) \tag{2.5}$$

where $\mathfrak{c}$ and γ are such that $\mathfrak{c} = \rho_0(\mathfrak{ab}) = \gamma \cdot \mathfrak{ab}$. They allow effective computations with ideal classes in $\mathcal{O}$, and are, henceforth, basic building blocks of the algorithms in this paper. Proposition 2.4 implies the run-time bounds for $*$ given in the following corollary.

Corollary 2.5 *There is an algorithm for the computation of the operation $*$ with input and output given in standard representation that executes in quadratic time.*

Proof This follows from the multiplication formulae given, e.g., in [Len82], and Proposition 2.4. ∎

2.4 Infrastructure

For the rest of this chapter we will assume $\Delta > 0$.

Let $\mathfrak{a}$ be reduced. Denote the set of reduced ideals in the class of $\mathfrak{a}$ by $C(\mathfrak{a})$. The reduction map allows us to traverse $C(\mathfrak{a})$ starting from $\mathfrak{a}$. See below for details. Here we define the "inverse reduction" map which is defined only on $\mathcal{R}_\Delta$, and which will allow us to traverse $C(\mathfrak{a})$ in the opposite direction. Let

$$\gamma'(\mathfrak{a}) = \gamma(\mathfrak{a}^\sigma)^\sigma, \quad \text{and} \quad \rho^{-1}(\mathfrak{a}) = \mathfrak{a}/\gamma'(\mathfrak{a}). \tag{2.6}$$

The notation is justified by the following lemma which is easily checked.

Lemma 2.6 *For $\mathfrak{a} \in \mathcal{R}_\Delta$, we have $\gamma'(\rho(\mathfrak{a})) = 1/\gamma(\mathfrak{a})$.* ∎

Indeed, Lemma 2.6 implies

$$\rho^{-1}(\rho(\mathfrak{a})) = \rho(\mathfrak{a})/\gamma'(\rho(\mathfrak{a})) = \mathfrak{a}/\gamma(\mathfrak{a}) \cdot \gamma(\mathfrak{a}) = \mathfrak{a}.$$

Assume $\Delta > 0$. The (logarithmic) "length" of a field element α is given by

$$\operatorname{Log} \alpha = \frac{1}{2} \log \left| \frac{\alpha^\sigma}{\alpha} \right|. \tag{2.7}$$

Obviously we have $\operatorname{Log}(\alpha\beta) = \operatorname{Log}(a) + \operatorname{Log}(\beta)$. Length is invariant under multiplication by elements of $\mathbb{Q}^*\langle 1, \sqrt{\Delta} \rangle$.

The lengths of reducing numbers of reduced ideals lie in a small range circumscribed by the two following easily proved, but important lemmata.

Lemma 2.7 $1/\sqrt{\Delta} < -\operatorname{Log}(\gamma(\mathfrak{a})) \leq \frac{1}{2} \log \Delta$. ∎

Lemma 2.8 $\log 2 \leq -\operatorname{Log}(\gamma(\mathfrak{a})\gamma(\rho(\mathfrak{a})))$. ∎

The proofs are elementary. See [Len82].

It is this narrow variation limited by a factor which is logarithmic in the discriminant that allowed Shanks to generalize his baby-step giant-step method from groups to the infra-structure of a real-quadratic order.

Definition 2.9 The sequence of $\mathcal{O}$-ideals $(\mathfrak{a}_d)_{d\in\mathbb{N}}$ with primitive $\mathfrak{a} = \mathfrak{a}_0 \in I_\Delta$ and $\mathfrak{a}_{d+1} = \rho(\mathfrak{a}_d)$ for all $d \geq 0$ is called the *reduction sequence* starting at $\mathfrak{a}$.

Let $\mathfrak{a} \in I_\Delta$, and $n \in \mathbb{N}$. We will use the notation $B(\mathfrak{a}, n)$ for the sequence $(\mathfrak{a}, \rho(\mathfrak{a}), \ldots, \rho^n(\mathfrak{a}))$ starting at $\mathfrak{a} \in \mathcal{R}_\Delta$, and $\mathring{B}(\mathfrak{a}, n)$ for its subsequence starting at index 1. For $n \in \mathbb{Z}_{<0}$, we will likewise write $B(\mathfrak{a}, n)$ for the sequence $(\mathfrak{a}, \rho^{-1}(\mathfrak{a}), \ldots, \rho^{-n})$, and $\mathring{B}(\mathfrak{a}, n)$ for its subsequence starting at -1. $B(\mathfrak{a}, 0) = \{\mathfrak{a}\}$, and $\mathring{B}(\mathfrak{a}, 0)$ is defined to be the empty set.

We assign two further sequences to a reduction sequence $(\mathfrak{a}_d)$: the sequence of relative generators γ_d, and a sequence of generators α_d relative to $\mathfrak{a}$. They are defined by

$$\gamma_d = \begin{cases} 1/\gamma(\mathfrak{a}_{d-1}) & \text{if } d > 0, \\ 1 & \text{if } d = 0, \end{cases} \tag{2.8}$$

which implies $\mathfrak{a}_d = \gamma_d \cdot \mathfrak{a}_{d-1}$, and

$$\alpha_d = \begin{cases} \prod_{i=1}^{d} \gamma_i & \text{if } d > 0, \\ 1 & \text{if } d = 0, \end{cases} \tag{2.9}$$

which implies $\mathfrak{a}_d = \alpha_d \cdot \mathfrak{a}_0$.

Lemma 2.10 *Let $(\mathfrak{a}_d)$ be a reduction sequence starting at some reduced $\mathfrak{a} = \mathfrak{a}_0$. Let $r_d = \operatorname{Log} \alpha_d$ for any $d > 0$. Then*

$$d \cdot \log 2 \leq r_{2d} \leq d \cdot \log \Delta, \qquad r_d > r_{d-1} > 0. \tag{2.10}$$

Proof This follows from Lemmata 2.7 and 2.8. ∎

The sequence (r_d) with $r_d = \operatorname{Log} \alpha_d$ will be called the *length sequence* of $(\mathfrak{a}_d)$, and r_d the *total length* of $B(\mathfrak{a}, d)$. The latter will also be denoted by $r(\mathfrak{a}, d)$. Finally, we extend $r(\mathfrak{a}, \cdot)$ to negative arguments by setting $r(\mathfrak{a}, -d) = -r(\rho^{-d}(\mathfrak{a}), d)$.

The set $\mathcal{C}$ of all reduced ideals in a given class can be mapped onto a discrete subset of the circle group $\mathbb{R}/R_\Delta\mathbb{Z}$ by taking the Log of relative generators relative to a fixed reduced representative $\mathfrak{a}$. Reduction cycles through $\mathcal{C}$. The set $\mathcal{C}$ is thus also called a cycle, or, in this case the cycle of $\mathfrak{a}$, and denoted by $\mathcal{C}(\mathfrak{a})$. The following statements give some details about this notion.

Lemma 2.11 *Let* $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$. *Assume* $\mathfrak{a} = \lambda \cdot \mathfrak{b}$ *with* $0 < \operatorname{Log} \lambda$. *Then* $\operatorname{Log} \lambda \geq -\operatorname{Log} \gamma(\mathfrak{b})$.

Look at the cycle $C(\mathcal{O})$, called the *principal cycle*. Arrange the ideals on this cycle on a circle according to the length of their generators taken mod R_Δ. The following lemma claims that no two ideals sit on the same spot on the circle.

Lemma 2.12 *Let* $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$, *and* $\lambda \in \mathcal{K}$. *If* $\lambda \cdot \mathfrak{a} = \mathfrak{b}$, *and* $\operatorname{Log} \lambda = 0$, *then* $\mathfrak{a} = \mathfrak{b}$.

PROOF $\operatorname{Log} \lambda = 0$ implies $\lambda \in \mathbb{Q}^* \langle 1, \sqrt{\Delta} \rangle$. If $\lambda \in \mathbb{Q}^*$, then the primitiveness of both $\mathfrak{a}$ and $\mathfrak{b}$ implies $\lambda = 1$. If $\lambda \in \mathbb{Q}^* \cdot \sqrt{\Delta}$, then, as can easily be seen by looking at the exponents of the ideals, $\mathfrak{a}$ and $\mathfrak{b}$ cannot both be reduced. ∎

Proposition 2.13 *For each* $\mathfrak{a} \in \mathcal{R}_\Delta$, *there exists some* $n \in \mathbb{N}$ *and a sequence* $1 = \alpha_0, \alpha_1, \ldots, \alpha_n \in \mathcal{K}$ *such that*

1. $\alpha_{i+1} \cdot \mathfrak{a} = \rho(\alpha_i \cdot \mathfrak{a})$ *for* $0 \leq i < n$.
2. $0 < \operatorname{Log} \alpha_i < \operatorname{Log} \alpha_j \leq R_\Delta$ *if* $0 < i < j \leq n$,
3. $C(\mathfrak{a}) = \{ \alpha_i \cdot \mathfrak{a} \mid 0 < i \leq n \}$,

If $\alpha_1, \ldots, \alpha_n$ *is such a sequence, and* $\alpha_n > 0$, *then* $1/\alpha_n$ *is the fundamental unit of* $\mathcal{O}$. ∎

Indeed, we have just constructed a sequence with these properties.

From Proposition 2.13 we obtain a generalization of Lemma 2.11.

Corollary 2.14 *Let* $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$. *Assume* $\mathfrak{a} = \lambda \cdot \mathfrak{b}$ *with* $0 < \operatorname{Log} \lambda$. *Let* (α_i) *be a sequence of numbers in* $\mathcal{K}$, *with properties as given in Proposition 2.13. Assume* $\mathfrak{a} = \alpha_k \cdot \mathfrak{b}$. *Then* $\operatorname{Log} \lambda \geq \operatorname{Log} \alpha_k$. ∎

Corollary 2.15 *Let* $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$, *and* $\lambda \in \mathcal{K}$ *with* $\mathfrak{b} = \lambda \cdot \mathfrak{a}$. *If* $0 < \operatorname{Log} \lambda \leq r(\mathfrak{a}, e)$ *for some* $e \in \mathbb{N}$, *then there exists* $f \in \mathbb{N}$ *with* $0 < f \leq e$ *and* $\mathfrak{b} = \rho^f(\mathfrak{a})$. ∎

Distance properties of ρ_0 are given in [Len82]. We collect them in the following proposition.

Proposition 2.16 *Let* $\mathfrak{a} \in I_\Delta$ *be an invertible* $\mathcal{O}$*-ideal. Let* $\mathfrak{c} = \mathfrak{a}_d$ *be the first reduced member of the reduction sequence starting at* $\mathfrak{a}$, *i.e.* $\rho_0(\mathfrak{a}) = \mathfrak{a}_d$. *Let further* $\gamma = \alpha_d$ *be the generator of* $\mathfrak{a}_d$ *relative to* $\mathfrak{a}$ *defined by* (2.9). *Then*

$$|\operatorname{Log}(\gamma)| < \tfrac{1}{2} \log(1 + \theta\sqrt{\Delta}) < \tfrac{1}{2} \log \Delta, \quad \text{where } \theta = \frac{1+\sqrt{5}}{2}. \tag{2.11}$$

Moreover, for $\gamma_1 = \gamma\gamma(\mathfrak{c})^{-1}\gamma(\rho(\mathfrak{c}))^{-1}$, *and* $\gamma_2 = \gamma\gamma'(\mathfrak{c})^{-1}\gamma'(\rho^{-1}(\mathfrak{c}))^{-1}$ *we have*

$$\rho^2(\mathfrak{c}) = \gamma_1 \cdot \mathfrak{a}, \quad \textit{with } \operatorname{Log}\gamma_1 \geq 0, \tag{2.12}$$

$$\rho^{-2}(\mathfrak{c}) = \gamma_2 \cdot \mathfrak{a}, \quad \textit{with } \operatorname{Log}\gamma_2 \leq 0. \tag{2.13}$$

■

Corollary 2.17 *For any* $\mathfrak{a} \in I_\Delta$, *there are* $\gamma_1, \gamma_2 \in \mathcal{K}$ *with* $\gamma_i \cdot \mathfrak{a} \in \mathcal{R}_\Delta$ *and*

$$-\tfrac{1}{2}\log\Delta < \operatorname{Log}(\gamma_1) < 0 < \operatorname{Log}(\gamma_2) < \tfrac{1}{2}\log\Delta. \tag{2.14}$$

Proof This corollary follows immediately from Proposition 2.16. ■

Lemma 2.18 *Let* $\mathfrak{a} \in I_\Delta$ *be a primitive ideal. If* $\mathfrak{c} = \gamma \cdot \mathfrak{a} \in \mathcal{R}_\Delta$ *with* $|\operatorname{Log}\gamma| < \frac{1}{2}\log\Delta$, *in particular if* $(\mathfrak{c}, \gamma) = \textsc{Rho0}(\mathfrak{a})$, *then* $H(\gamma) < \Delta^{3/4}$. *If* $\mathfrak{a}$ *has norm smaller than* Δ, *and in particular if* $\mathfrak{a}$ *is the product of two reduced ideals, then* $d(\gamma) < \Delta$.

Proof Since $\mathrm{N}(\mathfrak{c}) < \sqrt{\Delta}$, the ideal $\mathfrak{a}$ is integral and $\mathfrak{c} = \gamma\mathfrak{a}$, we have $|\mathrm{N}(\gamma)| < \sqrt{\Delta}$, too. On the other hand, $|\operatorname{Log}\gamma| < \frac{1}{2}\log\Delta$ implies

$$\frac{1}{\Delta} < \left|\frac{\gamma^\sigma}{\gamma}\right| < \Delta$$

Taken together, the inequalities yield $|\gamma|, |\gamma^\sigma| < \Delta^{3/4}$.

Moreover, $\mathfrak{c} = \gamma\mathfrak{a}$ implies that $\mathrm{N}(\mathfrak{a})(\gamma)$ is integral. Hence, the denominator of γ is bounded by Δ. ■

2.5 Auxiliary algorithms

In this section we give the properties of a number of very simple auxiliary algorithms that are employed in later chapters for ease of writing. The listings can be found at the end of this chapter.

The first two algorithms, CloseL and CloseR compute some γ_1, or some γ_2, respectively, that satisfy (2.14) given any $\mathfrak{a} \in I_\Delta$.

Lemma 2.19 *Given* $\mathfrak{a} \in I_\Delta$ *as input,* CloseL (CloseR) *returns* $\mathfrak{c}_i \in \mathcal{R}_\Delta$ *and* γ_i, *where* $i = 1$ *(or* $i = 2$, *respectively) such that* $\mathfrak{c}_i = \gamma_i \cdot \mathfrak{a}$ *and* γ_i *satisfy* (2.14). *Moreover,* $\operatorname{Log}\gamma_1 + r(\mathfrak{c}, 2) \geq 0$, *and* $\operatorname{Log}\gamma_2 + r(\mathfrak{c}, -2) \leq 0$. *The time complexity of both algorithms applied to an ideal with exponent* $a = O(\Delta^e)$ *for some fix* e *is bounded by* $O(\log^2\Delta)$.

We continue by giving simple variations, called Left and Right, that find the reduced ideal (equal to or) immediately to the left or right of a given $\mathfrak{a} \in I_\Delta$, and an algorithm EnumR that lists the reduced ideals in a section of length $\log\Delta$ on the right of $\mathfrak{a} \in I_\Delta$. We state the properties of the listed algorithms in the following two lemmata. They follow easily from Proposition 2.16.

Lemma 2.20 *Given* $\mathfrak{a} \in I_\Delta$ *as input,* Left (Right) *returns* $\mathfrak{c}_i \in \mathcal{R}_\Delta$ *and* γ_i*, where* $i = 1$ *(or* $i = 2$*, respectively) such that* $\mathfrak{c}_i = \gamma_i \cdot \mathfrak{a}$*. The* γ_i *satisfy*

$$\operatorname{Log}\gamma_1 \leq 0 \qquad \operatorname{Log}(\gamma_1/\gamma(\mathfrak{c}_1)) > 0 \tag{2.15}$$

$$\operatorname{Log}\gamma_2 \geq 0 \qquad \operatorname{Log}(\gamma_2/\gamma'(\mathfrak{c}_2)) < 0 \tag{2.16}$$

The time complexity of both algorithms applied to an ideal with exponent $a = O(\Delta^e)$ *is for fix* e *bounded by* $O(\log^2 \Delta)$*.*

Lemma 2.21 *Given* $\mathfrak{a} \in I_\Delta$ *with exponent* a *as input,* EnumR *returns*

$$S = \{\, (\mathfrak{a}, c) \mid \mathfrak{c} = \gamma \cdot \mathfrak{a} \text{ reduced, } 0 \leq \operatorname{Log}\gamma < \tfrac{1}{2}\log\Delta \,\} \tag{2.17}$$

The time complexity of EnumR *applied to an ideal with exponent* $a = O(\Delta^e)$ *is fox fix* e *bounded by* $O(\log^2 \Delta)$*.*

Length checks of the form $\operatorname{Log}\gamma \stackrel{?}{<} \delta$ in CloseL, CloseR, Left, and Right can be performed exactly without computing logarithms in order to avoid precision problems by transforming γ into standard representation $a + b\omega$. The sign of $\operatorname{Log}\gamma$ can now be determined by comparing the signs of a and b, if $\Delta \equiv 0 \pmod 4$, or comparing the signs of $(a + b/2)$ and b, if $\Delta \equiv 1 \pmod 4$.

Note that γ has to have small height for this procedure to be efficient. The relative generator γ does have small height if the ideal $\mathfrak{a}$ in the input of said algorithms has small norm, e.g., if it it is the product of two reduced ideals.

Finally, DistR is a simple-minded algorithm that computes elements of $\mathcal{O}$ that have given "length".

Lemma 2.22 *Given* $\delta \in \mathbb{R}_{>0}$*,* DistR *computes* $\alpha \in \mathcal{O}$ *satisfying* $\delta \leq \operatorname{Log}\alpha < \delta + \frac{1}{2}\log\Delta$ *in time* $O(\delta/\log\Delta \cdot \mu(\log\Delta)))$*.*

In both EnumR and DistR, the computation of logarithms can be avoided and the length checks in EnumR and DistR can be executed by exact computation, if $\delta \in \operatorname{Log}\mathcal{O}^*$, or if δ is a multiple of $\log\Delta$.

2.6 Computing Log

In this section we will see how Log can be evaluated with pre-determined precision on arguments which are quadratic numbers given in any of the representations introduced in Section 2.2. This can be done by employing Brent's algorithm [Bre76] for the computation of the logarithm using the AGM.

Fix an error bound δ. Let α be given in standard representation. Then we can apply Brent's algorithm directly to $|\alpha^\sigma/\alpha|$ using error bound δ. Call this algorithm Log.

We extend Log to arguments in compact representation. This is done by computing

$$\textsc{Log}(\alpha, \delta) = \textsc{Log}(\gamma, \delta/2) + \sum_{j=1}^{k} 2^{k-j} \textsc{Log}(\gamma_j, \delta/2^{k-j+1})$$

Analogously, we extend Log to arguments in power product representation.

Let $\mu(n)$ denote the time necessary for the multiplication of two n-bit natural numbers. Define the height of an integral quadratic number given in standard representation to be $\max(|\alpha|, |\alpha^\sigma|)$, and denote it by $H(\alpha)$.

Lemma 2.23 *Given an error bound $\delta > 0$ and $\alpha \in \mathcal{O}$ in standard representation with height bounded by H,* Log *computes $L \in \mathbb{Q}$ such that $0 \leq L - \operatorname{Log} \alpha < \delta$ using time and space bounded by $O^\sim(\log\log H + \mu(-\log \delta))$.*

Given likewise an error bound $\delta > 0$ and $\alpha \in \mathcal{O}$ in compact representation $(\gamma, \gamma_1, \ldots, \gamma_k)$ with components bounded in height by H, Log *computes $L \in \mathbb{Q}$ such that $0 \leq L - \operatorname{Log} \alpha < \delta$ using time and space bound by $O^\sim\big(k \cdot (\log\log H + \mu(k - \log \delta))\big)$.*

Given finally error bound $\delta > 0$ and $\alpha \in \mathcal{O}$ in power product representation $\big((\gamma, a_\gamma)\big)$ of length m where each γ has no more than K components with height bounded by H, and the a_γ are uniformly bounded by M, Log *computes $L \in \mathbb{Q}$ such that $0 \leq L - \operatorname{Log} \alpha < \delta$ using time bounded by $O^\sim(m \cdot K \cdot (\log\log H + \mu(\log m + \log M + K - \log \delta))$.*

Proof The proof of the first statement is contained in [Bre76]. See also [BB87]. The second and third statements follow immediately. ∎

Algorithm 2.1: Find reduced ideal closely on the right of $\mathfrak{a}$

Input: $\mathfrak{a} \in I_\Delta$

Output: $(\mathfrak{c}, \gamma)$, where $\mathfrak{c}$ is reduced, $\mathfrak{c} = \gamma \cdot \mathfrak{a}$, and $0 < \operatorname{Log}\gamma < \frac{1}{2}\log\Delta$

CloseR($\mathfrak{a}$)
1. Reduce the ideal $\mathfrak{b}$, until $\mathfrak{c} = \rho_0(\mathfrak{b})$ is obtained: $(\mathfrak{c}, \gamma) \leftarrow$ Rho0($\mathfrak{a}$).
2. **while** $\operatorname{Log}\gamma \le 0$
3. Let $\gamma \leftarrow \gamma/\gamma(\mathfrak{c})$, and $\mathfrak{c} \leftarrow \rho(\mathfrak{c})$.
4. **return** $\mathfrak{c}$ and γ.

Algorithm 2.2: Find reduced ideal closely on the left of $\mathfrak{a}$

Input: $\mathfrak{a} \in I_\Delta$

Output: $(\mathfrak{c}, \gamma)$, where $\mathfrak{c}$ is reduced, $\mathfrak{c} = \gamma \cdot \mathfrak{a}$, and $0 > \operatorname{Log}\gamma > -\frac{1}{2}\log\Delta$

CloseL($\mathfrak{a}$)
1. Reduce the ideal $\mathfrak{b}$, until $\mathfrak{c} = \rho_0(\mathfrak{b})$ is obtained: $(\mathfrak{c}, \gamma) \leftarrow$ Rho0($\mathfrak{a}$).
2. **while** $\operatorname{Log}\gamma \ge 0$
3. Let $\gamma \leftarrow \gamma/\gamma'(\mathfrak{c})$, and $\mathfrak{c} \leftarrow \rho^{-1}(\mathfrak{c})$.
4. **return** $\mathfrak{c}$ and γ.

Algorithm 2.3: Find reduced ideal immediately on the left of $\mathfrak{a}$

Input: $\mathfrak{a} \in I_\Delta$

Output: $(\mathfrak{c}, \gamma)$, where $\mathfrak{c}$ is reduced, $\mathfrak{c} = \gamma \cdot \mathfrak{a}$, and $\operatorname{Log}\gamma = \min\{\operatorname{Log}\gamma' \mid \mathfrak{c}' = \gamma' \cdot \mathfrak{a}$ is reduced, and $\operatorname{Log}\gamma' < 0\}$

Left($\mathfrak{a}$)
1. Reduce the ideal $\mathfrak{b}$, until $\mathfrak{c} = \rho_0(\mathfrak{b})$ is obtained: $(\mathfrak{c}, \gamma) \leftarrow$ Rho0($\mathfrak{a}$).
2. **while** $\operatorname{Log}\gamma \ge 0$
3. Let $\gamma \leftarrow \gamma/\gamma'(\mathfrak{c})$, and $\mathfrak{c} \leftarrow \rho^{-1}(\mathfrak{c})$.
4. **while** $\operatorname{Log}\gamma/\gamma(\mathfrak{c}) < 0$
5. Let $\gamma \leftarrow \gamma/\gamma(\mathfrak{c})$, and $\mathfrak{c} \leftarrow \rho(\mathfrak{c})$.
6. **return** $\mathfrak{c}$ and γ.

Algorithm 2.4: Find reduced ideal immediately on the right of $\mathfrak{a}$

Input: $\mathfrak{a} \in I_\Delta$

Output: $(\mathfrak{c}, \gamma)$, where $\mathfrak{c}$ is reduced, $\mathfrak{c} = \gamma \cdot \mathfrak{a}$, and $\operatorname{Log}\gamma = \min\{\operatorname{Log}\gamma' \mid \mathfrak{c}' = \gamma' \cdot \mathfrak{a} \text{ is reduced, and } \operatorname{Log}\gamma' < 0\}$

LEFT($\mathfrak{a}$)
1. Reduce the ideal $\mathfrak{b}$, until $\mathfrak{c} = \rho_0(\mathfrak{b})$ is obtained: $(\mathfrak{c}, \gamma) \leftarrow$ RHO0($\mathfrak{a}$).
2. **while** $\operatorname{Log}\gamma \le 0$
3. Let $\gamma \leftarrow \gamma/\gamma(\mathfrak{c})$, and $\mathfrak{c} \leftarrow \rho(\mathfrak{c})$.
4. **while** $\operatorname{Log}\gamma/\gamma'(\mathfrak{c}) > 0$
5. Let $\gamma \leftarrow \gamma/\gamma'(\mathfrak{c})$, and $\mathfrak{c} \leftarrow \rho^{-1}(\mathfrak{c})$.
6. **return** $\mathfrak{c}$ and γ.

Algorithm 2.5: Enumerate reduced ideals on the right of $\mathfrak{a}$

Input: $\mathfrak{a} \in I_\Delta, \delta \in \operatorname{Log}\ \mathcal{O}^*$

Output: $S = \{(\mathfrak{a}, c) \mid \mathfrak{c} = \gamma \cdot \mathfrak{a} \text{ reduced}, 0 \le \operatorname{Log}\gamma < \delta\}$

ENUMR($\mathfrak{a}$)
1. $(\mathfrak{c}, \gamma) \leftarrow$ RIGHT($\mathfrak{a}$).
2. $S \leftarrow \{(\mathfrak{c}, \gamma)\}$.
3. **while** $\operatorname{Log}\gamma < \delta$
4. $\gamma = \gamma/\gamma(\mathfrak{c}), \mathfrak{c} \leftarrow \rho(\mathfrak{c})$.
5. $S \leftarrow S \cup \{(\mathfrak{c}, \gamma)\}$.
6. **return** S.

Algorithm 2.6: Compute quadratic integer of given length

Input: Length $\delta \in \mathbb{R}_{>0}$

Output: $\alpha \in \mathcal{O}$ such that $\delta \le \operatorname{Log}\alpha < \delta + \frac{1}{2}\log\Delta$

DISTR(δ)
1. $\alpha \leftarrow 1, \mathfrak{a} \leftarrow \mathcal{O}$. **while** $\operatorname{Log}\alpha < \delta$
2. $\alpha \leftarrow \alpha/\gamma(\mathfrak{a}), \mathfrak{a} = \rho(\mathfrak{a})$.
3. **return** α.

Kapitel 3

Deterministic Algorithms

In this chapter we describe deterministic algorithms for the Regulator and Extended Discrete Logarithm Problem in *real quadratic fields*. See Section 1.3 for previous work. As noted there, all algorithms given build on an idea of Daniel Shanks for the computation of the class group of quadratic orders in [Sha71].

We give an overview of the material presented in this chapter.

In the first two sections we give a brief introduction to the general baby-step giant-step idea, and the variation proposed by Terr in [Ter00] as they apply in particular to the computation of discrete logarithms.

In the third section we explain how the concept of baby and giant step sequences can be appropriately defined in the infrastructure of a real quadratic order. We show under which condition, and after how many steps a giant step sequence meets a baby step sequence on the same cycle of reduced ideals.

In the fourth section we turn this knowledge into an algorithm for the computation of the regulator of a real quadratic order.

The fifth section deals with the EDLP. Again we will build on the results of the third section. The algorithm for the EDLP involves slightly more elaborate record keeping than the regulator algorithm.

Both algorithms are also given in variants that execute only exact integer operations.

3.1 The baby-step giant-step strategy

This section recalls the basic baby-step giant-step strategy. A good reference for the material here is [BJT97].

The application of Shanks's idea in its original form to the Element Order Problem is based on the following simple lemma.

Lemma 3.1 *Let n and B be two natural numbers with $n < B^2$. Then there exist natural numbers $0 \leq e, f < B$ with $n = eB + f$.* ∎

Assume we are given some $g \in G$, and some $B \in \mathbb{N}$ with $n = \operatorname{ord} g < B^2$. Then the previous lemma assures the existence of $e, f < B$ with $g^{eB} = g^{-f}$, and $n = eB + f$.

Lemma 3.2 *Let $g \in G$, and assume that $\operatorname{ord} g < B^2$. Then there exist integral $0 \leq e, f < B$ such that $g^{eB} = g^{-f}$. If (e, f) is minimal among all such pairs in the lexicographical ordering, then $\operatorname{ord} g = eB + f$.* ∎

Thus one proceeds as follows.

1. Initialize a table T_{baby} with size B and first entry $(1, 0)$.
2. Compute g^{-f} for $f = 1, \ldots, B-1$, and enter (g^{-f}, f) into T_{baby}, unless g^{-f} equals 1, in which case one prints f, and exits.
3. Compute powers g^{eB} for $e = 1, \ldots, B-1$, and, after each computation, look up the resulting element in table T_{baby}. If a match is found, say $g^{eB} = g^{-f}$, then print $eB + f$.

The algorithm executes two inversions, $\lfloor (\operatorname{ord} g)/B \rfloor$ table look-ups, and between a minimum of $\operatorname{ord} g - 1$, and a maximum of $B - 2 + \lfloor (\operatorname{ord} g)/B \rfloor < 2B - 3$ group operations.

The disadvantage of this algorithm lies in the requirement that the bound B must be known in advance, and should be close to $\sqrt{\operatorname{ord} g}$.

In order to avoid this problem, two variations where proposed in [BJT97], and [Ter00]. We sketch Terr's variation. It is based on a modification of Lemma 3.1.

Lemma 3.3 *For every $n \in \mathbb{N}$ there exists a unique pair of natural numbers $0 \leq f \leq e \leq \lfloor \sqrt{2n} \rfloor$ with $n = e(e+1)/2 + f$.* ∎

Assume we are again given some $g \in G$. The previous lemma assures the existence of natural $0 \leq f \leq e$ with $g^{e(e+1)/2} = g^{-f}$.

Lemma 3.4 *Let $g \in G$. Then there exists a unique pair $0 \leq f \leq e \leq \lfloor \sqrt{2 \operatorname{ord} g} \rfloor$ such that $g^{e(e+1)/2} = g^{-f}$. For this pair we have $\operatorname{ord} g = e(e+1)/2 + f$.* ∎

Terr's algorithm alternates baby and giant steps. In the e^{th} round the baby step table T_{baby} contains the powers g^{-f} for $0 \leq f \leq e$. Thus the eventual size of the baby step table is not known beforehand, but does not exceed $\lfloor \sqrt{2n} \rfloor$.

T_{baby} is initialized with the entry $(1, 0)$. Compute g^{-1}, set $e = 1$, and repeat the following steps.

1. Compute $g^{-e} = g^{1-e} \cdot g^{-1}$. Insert (g^{-e}, e) into T_{baby}.
2. Compute $g_e = g^{e(e+1)/2} = g^{e(e-1)/2}/g^{-e}$.
3. If $g_e = g^{-f} \in T_{\text{baby}}$, then print $e(e+1)/2 + f$.

In each round one multiplication, one division, and one table-lookup need to be performed. No more than $\sqrt{2 \operatorname{ord} g}$ rounds will be executed.

3.2 Baby-step giant-step algorithm for the Discrete Logarithm Problem

It is straightforward to extend the algorithm given in the previous section to the solution of the Discrete Logarithm Problem. Given $g, h \in G$, we seek a relation of the form $g^{e(e+1)/2} h^{-1} = g^{-f}$. Hence, we initialize the first element of the giant-step sequence g_0 with h^{-1}, and proceed as before.

Lemma 3.5 *Let $g, h \in G$. If $h \in \langle g \rangle$, then there exists a unique pair $0 \leq f \leq e \leq \lfloor \sqrt{2 \operatorname{ord} g} \rfloor$ such that $g^{e(e+1)/2} h^{-1} = g^{-f}$. Given this pair, the discrete logarithm of h with basis g is $n = e(e+1)/2 + f$.* ∎

In order to detect the case $h \notin \langle g \rangle$, we need to know the order of g. If this is not known in advance, we may compute two giant step sequences in parallel, one starting at 1, the second at h^{-1}. If the first one meets an element in the baby step table before the second one does, then we know that the Discrete Logarithm Problem has no solution.

Algorithm 3.1: General discrete logarithm algorithm

Description: Baby-step giant-step algorithm, Terr's variant. Detects $h \notin \langle g \rangle$.

Input: Group G, elements $g, h \in G$

Output: Smallest $n \in \mathbb{N}$ such that $g^n = h$, or "$h \notin \langle g \rangle$"

GenDL(g, h)

1. **if** $h = 1$ **then print** 0
2. *Initialize:* $T_{\text{baby}} \leftarrow \{(1, 0)\}, g_0 \leftarrow 1, e \leftarrow 1$.
3. **repeat**
4. Compute $g^{-e} = g^{1-e} \cdot g^{-1}$. Insert (g^{-e}, e) into T_{baby}.
5. Compute $g_e \leftarrow g_{e-1}/g^{-e}$, and $h_e \leftarrow g_e \cdot h^{-1}$.
6. **if** $h_e = g^{-f} \in T_{\text{baby}}$ **then print** $e(e+1)/2 + f$, and **exit**.
7. **if** $g_e = g^{-f} \in T_{\text{baby}}$ **then print** "$h \notin \langle g \rangle$" and **exit**.
8. $e \leftarrow e + 1$.

Proposition 3.6 *Given a group G, and two elements $g, h \in G$ such that $h \in \langle g \rangle$, the Algorithm 3.2, named* GenDL*, terminates and computes minimal $n \in \mathbb{N}$ such that $h = g^n$. In this case* GenDL *executes no more than $\lfloor \sqrt{2n} \rfloor$,*

and no less than $\lceil\sqrt{2n}\rfloor - 2$ *rounds. If* $h \notin \langle g \rangle$*, then* GenDL *terminates after no more than* $\lfloor\sqrt{2 \operatorname{ord} g}\rfloor$ *rounds and announces this fact.*

Proof Assume first that $h \in \langle g \rangle$. Let $n \in \mathbb{N}$ be minimal such that $h = g^n$, and $q = \max(\, k \mid k(k+1)/2 \leq n \,)$. Obviously, $n < \operatorname{ord} g$.

If $h = 1$, then GenDL terminates in step 1 and prints the correct result. If $h \neq 1$, then $n, q > 0$. Let $r = n - q(q+1)/2$. Obviously, $q \geq r \geq 0$, due to the maximality of q. Thus T_{baby} contains the pair (g^{-r}, r) in round q after step 4. Since $h_q = g^{q(q+1)/2}/h = g^{-r}$, GenDL detects a match in step 6 of the same round, and, hence, terminates here if not earlier. If it does terminate here, then it clearly prints the correct result.

We convince ourselves that GenDL does not terminate earlier. Assume GenDL terminates in step 6 in round e, finding, say, $h_e = g^{-f}$ for some $f \leq e$. Then $e(e+1)/2 + f \geq n$ due to the minimality of n, and, hence, $e \geq q$.

Assume GenDL terminates in step 7 in round e, finding, say, $g_e = g^{-f}$ for some $f \leq e$. Then $e(e+1)/2 + f \geq \operatorname{ord} g > n$. Thus $e \geq q$, and GenDL detects the match in step 6 first.

We have found that GenDL does terminate in round q if $h \in \langle g \rangle$. We clearly have $\lceil\sqrt{2n}\rfloor - 2 \leq q \leq \lfloor\sqrt{2n}\rfloor$.

Assume now that $h \notin \langle g \rangle$. Then GenDL can never exit in step 6. However, we see that GenDL does stop in round q in step 7 when $q = \max(\, k \mid k(k+1)/2 \leq \operatorname{ord} g \,)$ by the same reasoning as before. Thus it correctly announces "$h \notin \langle g \rangle$". ∎

Terr's baby-step giant-step variant has running time complexity $O(\sqrt{n})$, where n is the discrete logarithm of h with respect to g. It is possible to stay within this complexity, and reorganize the search so that it finds solutions with specific properties first instead of walking bottom-up. One possible such variant exposes e.g. solutions to the Discrete Logarithm Problem with small Hamming weight.

3.3 Baby-step giant-step ideal sequences

In the preceding two sections we have seen that in any group sequences of the form $1, g^{-1}, g^{-2}, \ldots, g^{-f}$, and $g, g^3, g^6, \ldots, g^{e(e-1)/2}$ eventually meet for any group element g. We have called these sequences baby-step and giant-step sequences.

In this section we will see that these sequences can be generalized to sequences in the group-like infrastructure of a real quadratic order. They will also be shown to necessarily meet. The match yields as an analog to the order of a group element the length of the relative generator of two members of the same cycle.

Here, the length of a field element is given by the value of the Log function introduced in Chapter 2. For background on the infrastructure of a real quadratic order see Chapter 2.

We will keep the notation of that chapter. In particular, we let $\mathcal{O} = \mathcal{O}_\Delta$ be a real quadratic order with discriminant Δ and field of fractions $\mathcal{K}$. We assume $\mathcal{K}$ is embedded into $\mathbb{R}$, and denote the non-trivial automorphism of $\mathcal{K}$ by σ. The group of invertible $\mathcal{O}$-ideals is denoted by I_Δ, and its subset of primitive integral reduced $\mathcal{O}$-ideals by $\mathcal{R}_\Delta$. Reduction sequences starting at reduced ideals $\mathfrak{a} \in \mathcal{R}_\Delta$ will be called *baby-step ideal sequences.*

We will now introduce the notion of a giant-step ideal sequence. The length of a generator of a giant-step ideal relative to its successor will be called the *width* of the next giant step. We want the successive widths to be close to, but smaller than some sequence of positive real numbers which we will call its spacing sequence.

Definition 3.7 Let $(s_d)_{d\in\mathbb{N}}$ be a sequence of real numbers. We say that a sequence $(\mathfrak{b}_d)$ of $\mathcal{O}$-ideals $\mathfrak{b}_d \in \mathcal{R}_\Delta$ is a *giant-step ideal sequence* relative to the spacing sequence (s_d) (or that (s_d) is the spacing sequence of the giant-step sequence $(\mathfrak{b}_d)$), if there exist for all $d > 0$ relative generators $\beta_d \in \mathcal{K}$ such that $\mathfrak{b}_d = \beta_d \cdot \mathfrak{b}_{d-1}$ with

$$s_d - \tfrac{1}{2}\log\Delta < -\operatorname{Log}\beta_d \le s_d. \tag{3.1}$$

Thus the width $-\operatorname{Log}\beta_d$ of giant step d is to equal the corresponding element s_d of the spacing sequence with an allowed aberration not larger than the maximal size of a single reduction step.

We give two trivial examples.

Lemma 3.8 *Let ϵ be the fundamental unit with $\operatorname{Log}\epsilon = R$. For any $\mathfrak{a} \in \mathcal{R}_\Delta$,*

1. *the constant sequence $(\mathfrak{a})$ is a giant-step sequence with constant spacing sequence (R) and $\beta_d = \epsilon$;*
2. *the "reverse baby-step" sequence $(\mathfrak{a}_d)$ with $\mathfrak{a}_d = \rho^{-d}(\mathfrak{a})$ is a giant-step sequence with constant spacing sequence (0) and $\beta_d = 1/\gamma'(\mathfrak{a}_{d-1})$.* ∎

Lemma 3.9 *For every spacing sequence (s_d) and $\mathcal{O}$-ideal $\mathfrak{b} \in \mathcal{R}_\Delta$, there is a giant-step sequence relative to (s_d) starting at $\mathfrak{b}$.*

Proof Let $\mathfrak{b}_0 = \mathfrak{b}$. For $d > 0$ we define $\mathfrak{b}_d$ and β_d iteratively.

Look at the baby-step sequence $(\mathfrak{a}_e)$ with $\mathfrak{a}_0 = \mathfrak{b}_d$. Let α_e be its sequence of generators relative to $\mathfrak{b}_{d-1}$, and r_e its length sequence. By Lemma 2.7 we have $r_{e+1} - r_e < \frac{1}{2}\log\Delta$, and by Lemma 2.8 the sequence r_e tends to $-\infty$ if e tends to $-\infty$, and to ∞ if e tends to ∞.

Hence there is an $e \in \mathbb{Z}$ with $s_d - \frac{1}{2}\log\Delta < -r_e \le s_d$. Set $\mathfrak{b}_d = \rho^e(\mathfrak{b}_{d-1})$, and $\beta_d = \alpha_e$. It is obvious that β_d satisfies (3.1). ∎

Lemma 3.10 *If $(\mathfrak{b}_d)$ is a giant sequence relative to some spacing sequence (s_d) bounded from below by $\frac{1}{2}\log\Delta$, and β_d relative generators with size prescribed by* (3.1)*, then*

$$\operatorname{Log}\beta_d \le \operatorname{Log}\gamma(\mathfrak{b}_d) < 0 \tag{3.2}$$

for all $d > 0$, and $\lim_{k\to\infty}\sum_{i=1}^{k}\operatorname{Log}\beta_i \to -\infty$.

PROOF The first claim follows from Lemma 2.11 applied to $\mathfrak{b}_{d-1} = 1/\beta_d \cdot \mathfrak{b}_d$. Here we use that the inequalities $s_d \ge \frac{1}{2}\log\Delta$ and (3.1) imply $\operatorname{Log}(1/\beta_d) > 0$. Thus each giant step is at least as wide as an inverse reduction step.

The second statement follows from Lemma 2.8, since the latter implies that the sum of the widths of two consecutive giant steps exceeds $\log 2$. This is indeed clear if each of the two giant steps involves just one inverse reduction. If one of them does not, then it is individually at least as large as $\log 2$ due to Corollary 2.14 and (2.10). ∎

Lemma 3.11 *Suppose we are given two reduced ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$ in the same ideal class, and $\lambda \in \mathcal{K}$ with $\mathfrak{b} = \lambda\mathfrak{a}$ and $0 < \operatorname{Log}\lambda \le R$. Let $(\mathfrak{a}_d)$ be the baby-step sequence starting at $\mathfrak{a} = \mathfrak{a}_0$.*

Let (s_d) be some spacing sequence with $s_d \ge \frac{1}{2}\log\Delta$ for all $d \in \mathbb{N}$. Let $(\mathfrak{b}_d)$ be a giant-step sequence relative to (s_d) starting at $\mathfrak{b}$ with relative generators β_d satisfying (3.1)*. Define $\lambda_k = \lambda \cdot \prod_{i=1}^{k}\beta_i$ for all $k \ge 1$. Then the set $\{\, k \ge 1 \mid \operatorname{Log}\lambda_k \le s_{k+1} \,\}$ is not empty. Let e be its minimum. We have $\operatorname{Log}\lambda_e > 0$.*

PROOF From Lemma 3.10 we see that $\operatorname{Log}\lambda_k \to -\infty$, proving the first assertion of the lemma. Let e be as in the lemma. Assume $\operatorname{Log}\lambda_e \le 0$. Since $\operatorname{Log}\lambda_0 = \operatorname{Log}\lambda > 0$ we know that $e > 0$. Then (3.1) implies

$$\operatorname{Log}\lambda_{e-1} = \operatorname{Log}\lambda_e - \operatorname{Log}\beta_e \le s_e.$$

in contradiction to the minimality of e. ∎

From the preceding lemma we will now deduce the proposition that underlies the classical baby-step giant-step algorithm for the computation of the regulator of a quadratic order.

Proposition 3.12 *Suppose we are given two reduced ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$ in the same ideal class. Let $\lambda \in \mathcal{K}$ be such that $\mathfrak{b} = \lambda\mathfrak{a}$ and $0 < \operatorname{Log}\lambda \le R$. Let $(\mathfrak{a}_d)$ be the baby-step sequence starting at $\mathfrak{a} = \mathfrak{a}_0$ with sequence of generators (α_d), and $(\mathfrak{b}_d)$ a giant-step sequence relative to a spacing sequence (s_d) with constant $s_d = s$. Let $E_{\mathrm{S}} = \log_2\Delta + \sqrt{R/\log 2}$. If for any $F > 0$*

$$E_{\mathrm{S}}\log 2 - \tfrac{1}{2}\log\Delta < s \le r(\mathfrak{a}, F), \tag{3.3}$$

then there exist $e, f \in \mathbb{N}$ with $e \le \lfloor E_{\mathrm{S}} \rfloor$ and $0 < f \le F$ such that $\mathfrak{a}_f = \mathfrak{b}_e$. Moreover, for the lexicographically smallest such pair (e, f), we have $\operatorname{Log}\alpha_f - \sum_{i=1}^{e}\operatorname{Log}\beta_i = \operatorname{Log}\lambda$.

PROOF We retain data and assumptions of the proposition. Define λ_i as in Lemma 3.11. The lemma is applicable in the current set-up since $s > \frac{1}{2}\log\Delta$ by assumption (3.3). Let e be the minimum of $\{\, k \mid \operatorname{Log}\lambda_k \le s_{k+1} \,\}$. Then by Lemma 3.11 we have $R \ge \operatorname{Log}\lambda_e > 0$.

Let $E' = \lfloor E_{\mathrm{S}} \rfloor$. From (3.1) and the left-hand side of (3.3) we obtain

$$\begin{aligned}\operatorname{Log}\lambda_{E'} &= \operatorname{Log}\lambda + \sum_{i=1}^{E'} \operatorname{Log}\beta_i \\ &\le \operatorname{Log}\lambda - E'(s - \tfrac{1}{2}\log\Delta) < \operatorname{Log}\lambda - R \le 0.\end{aligned}$$

Hence $e < E_{\mathrm{S}}$. From Corollary 2.15 and $0 < \operatorname{Log}\lambda_e \le s \le r(\mathfrak{a}, F)$, it follows that there exists some $f \in \mathbb{N}$ with $0 < f \le F$ and $\mathfrak{a}_f = \mathfrak{b}_e$.

For the smallest such f we have $R \ge \operatorname{Log}\alpha_f > 0$. Hence $\operatorname{Log}\alpha_f = \operatorname{Log}\lambda_e$ follows from the equality of the ideals. This proves the last claim of the proposition. ∎

Note that for any Δ and $\mathfrak{a}$ we have $E_{\mathrm{S}}\log 2 - \frac{1}{2}\log\Delta < r(\mathfrak{a}, 2\lceil E_{\mathrm{S}} \rceil)$ by Lemma 2.8. Hence we can apply the preceding proposition to find a giant-step sequence that meets a given baby-step sequence: Choose some integral $\bar{E} \ge E_{\mathrm{S}}$, enumerate the first $\bar{E}$ members of the baby-step ideal sequence starting at $\mathfrak{a}$, and then compute a giant-step sequence with constant spacing sequence $(r(\mathfrak{a}, \bar{E}))$. According to the preceding proposition, the giant-step sequence will meet the baby-step sequence after no more than E_{S} members are computed.

The first step of this algorithm presupposes, however, that an approximation to the regulator is available.

In order to treat the case that the approximate size of the regulator is not known beforehand, we will now give the analogue of Lemma 3.3 in the infrastructure.

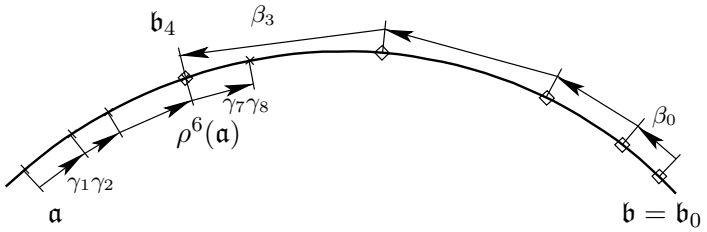


Abbildung 3.1: Baby step sequence $(\rho^d(\mathfrak{a}))$, and giant step sequence $(\mathfrak{b}_e)$ meet

Proposition 3.13 *Suppose we are given two reduced ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$ in the same ideal class. Let $\lambda \in \mathcal{K}$ be such that $\mathfrak{b} = \lambda\mathfrak{a}$ and $0 < \operatorname{Log}\lambda \le R$. Let*

$(\mathfrak{a}_d)$ be the baby-step sequence starting at $\mathfrak{a} = \mathfrak{a}_0$ with generators α_d relative to $\mathfrak{a}$, and $(\mathfrak{b}_d)$ be a giant-step sequence relative to the spacing sequence $s_d = \operatorname{Log} \alpha_{2(d+L)}$ starting at $\mathfrak{b}$ with relative generators β_d satisfying (3.1), where L is such that $\operatorname{Log} \alpha_{2L+2} \geq \frac{1}{2} \log \Delta$. Define $\lambda_k = \lambda \cdot \prod_{i=1}^{k} \beta_i$. Let finally $E_{\mathrm{T}} = \log_2 \Delta + \sqrt{2R/\log 2}$.

1. *If e is the minimum of $\{\, k \mid \operatorname{Log} \lambda_k \leq s_{k+1} \,\}$, then $e < \lceil E_{\mathrm{T}} \rceil$ and there exists some $f \in \mathbb{N}$ with $0 < f \leq 2(e + L + 1)$ such that $\mathfrak{a}_f = \mathfrak{b}_e$.*

2. *If $(e, f) \in \mathbb{N}^2$ is lexicographically the smallest pair such that $0 < f \leq 2(e + L + 1)$ and $\mathfrak{a}_f = \mathfrak{b}_e$, then $0 < \operatorname{Log} \alpha_f - \sum_{i=1}^{e} \operatorname{Log} \beta_i \leq R$.*

3. *Let $e \in \mathbb{N}$ be minimal such that there exists an $f \in \mathbb{N}$ with $0 < f \leq 2(e + L + 1)$ and $\mathfrak{a}_f = \mathfrak{b}_e$. If $e > 1$, then there is only one such f.*

4. *Assume $L > \frac{1}{2} \log_2 \Delta$, and $\mathfrak{a} = \mathfrak{b}$. If $e \in \mathbb{N}$ is again minimal such that there exists an $f \in \mathbb{N}$ with $0 < f \leq 2(e + L + 1)$ and $\mathfrak{a}_f = \mathfrak{b}_e$, then $\operatorname{Log} \alpha_{2(e+L+1)} > \sqrt{2R \log 2} - \frac{1}{2} \log 2$.*

Before proceeding to the proof of Proposition 3.13, we will introduce the notion of a giant-step sequence governed by a baby-step sequence which is prompted by the proposition.

Definition 3.14 Let $(\mathfrak{a}_d)$ be a baby-step sequence. We say that a sequence $(\mathfrak{b}_e)$ of $\mathcal{O}$-ideals $\mathfrak{b}_e \in \mathcal{R}_\Delta$ is a giant-step sequence governed by $(\mathfrak{a}_d)$ if it is a giant-step sequence relative to (s_d) with $s_d = \operatorname{Log} \alpha_{2(d+L)}$ for some L with $\operatorname{Log} \alpha_{2L+2} \geq \frac{1}{2} \log \Delta$.

Thus the preceding proposition says that a giant step sequence meets any baby step sequence on the same cycle if it is governed by it.

Note that for any baby-step sequence $(\mathfrak{a}_d)$ and any $\mathfrak{b} \in \mathcal{R}_\Delta$ it is easy to compute a giant-step sequence governed by $(\mathfrak{a}_d)$ and starting at $\mathfrak{b}$. Choosing $L = \frac{1}{2} \log_2 \Delta$ ensures $\operatorname{Log} \alpha_{2L+2} > \frac{1}{2} \log \Delta$. Set $\mathfrak{b}_0 = \mathfrak{b}$, and compute for any $e > 0$ the result $(\mathfrak{c}, \gamma)$ of CloseR with input $\mathfrak{b}_{e-1}/\mathfrak{a}_{2(e+L)}$. The ideal $\mathfrak{c}$ will have a generator $\beta_e = \gamma/\alpha_{2(e+L)}$ relative to $\mathfrak{b}_{e-1}$ that satisfies

$$\operatorname{Log} \alpha_{2(e+L)} - \tfrac{1}{2} \log \Delta < -\operatorname{Log} \beta_e < \operatorname{Log} \alpha_{2(e+L)}$$

due to Lemma 2.19. Thus we may set $\mathfrak{b}_e = \mathfrak{c}$.

Proof (of Proposition 3.13) Let $(\mathfrak{a}_d)$ be the baby-step sequence starting at $\mathfrak{a} \in \mathcal{R}_\Delta$, and $(\mathfrak{b}_e)$ be a giant-step sequence with spacing sequence $s_d = \operatorname{Log} \alpha_{2(d+L)}$ starting at $\mathfrak{b} \in \mathcal{R}_\Delta$, where α_d is defined by (2.9). Let $\lambda \in \mathcal{K}$ be such that $\mathfrak{b} = \lambda\mathfrak{a}$ and $0 < \operatorname{Log} \lambda \leq R$, and define $\lambda_k = \lambda \cdot \prod_{i=1}^{k} \beta_i$ for all $k \geq 1$.

Claim 1. In the given situation we can apply Lemma 3.11, since $s_d \geq s_1 = \operatorname{Log} \alpha_{2(L+1)} \geq \frac{1}{2} \log \Delta$ for all $d \geq 1$. Let $E = \lceil E_{\mathrm{T}} \rceil$, and $e = \min(\, k \mid \operatorname{Log} \lambda_k \leq s_{k+1} \,)$. Then we have by (3.1) and Lemma 2.8

$$\begin{aligned} \operatorname{Log} \lambda_E &= \operatorname{Log} \lambda + \sum_{i=1}^{E} \operatorname{Log} \beta_i \\ &\leq \operatorname{Log} \lambda - \sum_{i=1}^{E} (\operatorname{Log} \alpha_{2(i+L)} - \tfrac{1}{2} \log \Delta) \\ &\leq \operatorname{Log} \lambda + E/2 \cdot \log \Delta - \frac{E(E+1)}{2} \log 2 < \operatorname{Log} \lambda - R \leq 0. \end{aligned}$$

Hence $e < \lceil E_{\mathrm{T}} \rceil$. Apply Corollary 2.15 to $\mathfrak{b}_e = \lambda_e \cdot \mathfrak{a}$. Claim 1 of the proposition follows.

Claim 2. Let (e, f) be a pair of natural numbers with properties as given in the condition of claim 2. The equality $\mathfrak{a}_f = \mathfrak{b}_e$ implies $\operatorname{Log} \alpha_f = \operatorname{Log} \lambda_e + kR$. By definition $\operatorname{Log} \alpha_f > 0$, and $\operatorname{Log} \lambda_e < R$. Hence $k \geq 0$, and $\operatorname{Log} \alpha_f \geq \operatorname{Log} \lambda_e$.

Since $f \leq 2(e + L + 1)$ we also have $\operatorname{Log} \alpha_{2(e+L+1)} \geq \operatorname{Log} \lambda_e$. We find that $e \geq \min(\, k \mid \operatorname{Log} \lambda_k \leq \operatorname{Log} \alpha_{2(k+L+1)} \,)$. On the other hand $e \leq \min(\, k \mid \operatorname{Log} \lambda_k \leq \operatorname{Log} \alpha_{2(k+L+1)} \,)$ by claim 1 due to the minimality of e. Hence we have equality, and we conclude from Lemma 3.11 that $\operatorname{Log} \lambda_e > 0$.

Finally, if $\operatorname{Log} \alpha_f > R$ we could find some $0 < f' < f$ for which $\alpha_{f'} = \alpha_f \cdot \epsilon$ which would violate the minimality of f. Hence we obtain $k = 0$, and $\operatorname{Log} \lambda + \sum_{i=1}^{e} \operatorname{Log} \beta_i = \operatorname{Log} \lambda_e = \operatorname{Log} \alpha_f$. Now the size constraints for $\operatorname{Log} \lambda$ imply claim 2.

Claim 3. Suppose $e > 0$ is chosen as before and there exist some $d, f \in \mathbb{N}$ with $0 < d < f \leq 2(e + L + 1)$ and $\mathfrak{a}_d = \mathfrak{a}_f = \mathfrak{b}_e$. This implies that there exists some d' with $d \leq d' < f$ and $\mathfrak{a}_{d'} = \mathfrak{a}$. Then $\mathring{B}(\mathfrak{a}, d')$ contains the whole cycle, hence in particular $\mathfrak{b}_{e-1}$. Due to the minimality of e we have $\mathfrak{b}_{e-1} \notin \mathring{B}(\mathfrak{a}, 2(e + L))$. It follows that $d' = 2(e + L) + 1$, and $\mathfrak{b}_{e-1} = \mathfrak{a}_{d'} = \mathfrak{a}$. Since $0 < \operatorname{Log} \lambda_e < \operatorname{Log} \lambda_{e-1} \leq \operatorname{Log} \lambda_0 \leq R$ by Lemma 3.10, we conclude $\operatorname{Log} \lambda_{e-1} = \operatorname{Log} \lambda_0 = R$. Thus applying Lemma 3.10 once more we see that $e = 1$.

Claim 4. Assume $L > \frac{1}{2} \log_2 \Delta$, and $\mathfrak{a} = \mathfrak{b}$. Hence $\operatorname{Log} \lambda = R$, and $\operatorname{Log} \alpha_{2(i+L)} > \frac{1}{2} \log \Delta$ for all $i \in \mathbb{N}$ due to Lemma 2.8.

In the proof of claim 2 we have seen that $\operatorname{Log} \lambda + \sum_{i=1}^{e} \operatorname{Log} \beta_i = \operatorname{Log} \lambda_e =$

$\operatorname{Log}\alpha_f$. Thus

$$\begin{aligned} R &\le \operatorname{Log}\alpha_{2(e+L+1)} - \sum_{i=1}^{e} \operatorname{Log}\beta_i \\ &\le (e+1)\operatorname{Log}\alpha_{2(e+L+1)} - \sum_{i=1}^{e} \operatorname{Log}(\alpha_{2(e+L+1)}/\alpha_{2(i+L)}) \\ &\le (e+1)\operatorname{Log}\alpha_{2(e+L+1)} - \frac{e(e+1)}{2}\log 2. \end{aligned}$$

The first inequality follows from the monotony of $\operatorname{Log}\alpha_k$; the second from (3.1); and the third again from Lemma 2.8.

Now compute the minimum of the function $f(x) = R/(x+1)+x/2\cdot\log 2$. This yields the bound for $\operatorname{Log}\alpha_{2(e+L+1)}$ in claim 4. ∎

3.4 Fundamental Unit

In this section we will apply the results of Section 3.3 to the computation of the fundamental unit of a real quadratic order. We will continue to use the notation from the preceding section. The proposed algorithm will be called TerrUnit. In this section we will first describe TerrUnit, prove its correctness, give a bound for its run-time, and then give an example of its application.

TerrUnit determines a power product representation of the fundamental unit of $\mathcal{K}$ by computing a baby step sequence and and a giant step sequence governed by it both starting at the unit ideal. When they meet at $\mathfrak{a}_f = \mathfrak{b}_e$ we obtain the fundamental unit from the generator of $\mathfrak{a}_f$ and the relative generators β_d of the giant step sequence with $0 < d \le e$.

After the power product representation of the fundamental unit is obtained, we can apply Brent's algorithm for logarithm computation to each factor and obtain the regulator as the sum of the results. This can be done with low precision. Starting from a regulator approximation, it is then possible to obtain in polynomial time a compact representation of the fundamental unit, see [BTW95], Theorem 4.1. This may in turn be used to find an approximation with improved precision in time depending quadratically on the number of bits desired, see e.g. Maurer's results in [Mau00].

Thus TerrUnit proceeds as follows. First, initialize $\mathfrak{a}_0$, and $\mathfrak{b}_0$ to equal $\mathcal{O}$. Then continue iteratively. In the d-th iteration, do the following:

1. Compute baby step ideals $\mathfrak{a}_{2d+1}$, and $\mathfrak{a}_{2(d+1)}$.
2. If $\mathfrak{b}_d = \mathfrak{a}_f \in B(\mathfrak{a}, 2(d+1))$, then return ϵ given by

$$\epsilon = \alpha_f^{-1} \cdot \prod_{i=1}^{d} \beta_i \tag{3.4}$$

3. Compute $\mathfrak{b}_{d+1}$, and β_{d+1}. Store β_{d+1}. Set $d = d + 1$.

To check whether $\mathfrak{b}_d \in B(\mathfrak{a}, 2(d+1))$ in step 2 above, TerrUnit uses a table. Each table entry contains a baby step ideal $\mathfrak{a}_f$ together with its generator α_f. (In practice it is sufficient to store a pointer to $\mathfrak{a}_{f-1}$ instead of α_f in order to save space, since it is possible to compute a product expansion of α_f by inspecting the sequence $\mathfrak{a}_i, i = 0, \ldots, f-1$.)

A listing of TerrUnit can be found on the following page. It uses two sub-algorithms, BStep and GStep, given below.

Algorithm 3.2: Baby Step

Description: Adds two ideals to the baby-step table

Input: Baby step table S, its last element $\mathfrak{a}$ with generator α

Output: S, and the new last element $\mathfrak{a}'$ with generator α'

---

1. $\mathfrak{a}_1 \leftarrow \rho(\mathfrak{a}), \alpha_1 \leftarrow \alpha/\gamma(\mathfrak{a})$,
2. $\mathfrak{a}_2 \leftarrow \rho(\mathfrak{a}_1), \alpha_2 \leftarrow \alpha_1/\gamma(\mathfrak{a}_1)$,
3. $S \leftarrow S \cup \{ (\mathfrak{a}_i, \alpha_i) \mid i = 1, 2 \}$.
4. **return** $S, \mathfrak{a}_2, \alpha_2$.

Algorithm 3.3: Giant Step

Description: Giant step with width $\operatorname{Log} \alpha$ for some given $\alpha \in \mathcal{K}$

Input: $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$, and $\alpha \in \mathcal{K}$ such that $\mathfrak{a} = (\alpha)$

Output: $\mathfrak{b}'$ and β such that $\mathfrak{b}' = \beta \cdot \mathfrak{b}$ and α and β satisfy (3.1)

GStep$(\mathfrak{b}, \mathfrak{a}, \alpha)$

1. $(\mathfrak{b}', \gamma) \leftarrow$ CloseR$(\mathfrak{b}/\mathfrak{a}), \beta \leftarrow \gamma/\alpha$.
2. **return** $(\mathfrak{b}', \beta)$

We begin the analysis of TerrUnit by focusing on the giant step computation.

Lemma 3.15 *Given $\mathfrak{b}, \mathfrak{a} \in \mathcal{R}_\Delta$ and a generator α of $\mathfrak{a}$,* GStep *computes β and $\mathfrak{b}' = \beta \cdot \mathfrak{b}$ such that*

$$\operatorname{Log} \alpha - \tfrac{1}{2} \log \Delta < -\operatorname{Log} \beta \leq \operatorname{Log} \alpha, \tag{3.5}$$

is satisfied.

Algorithm 3.4: Deterministic Computation of Fundamental Unit

Description: Computation of a power product representation of the fundamental unit of order $\mathcal{O}$ with discriminant Δ

Input: Discriminant Δ

Output: Fundamental unit ϵ

TerrUnit(Δ)

1. *Initialization:*
   $\mathfrak{a}_0 \leftarrow \mathcal{O}, \mathfrak{b}_0 \leftarrow \mathcal{O}, \alpha_0 = 1, T \leftarrow \emptyset, d = 0, L \leftarrow \left\lceil \frac{1}{2} \log_2 \Delta \right\rceil$.
2. **for** $l \leftarrow 0$ **to** $L - 1$
3. *Baby step:*
   $(T, \mathfrak{a}_{2(l+1)}, \alpha_{2(l+1)}) = \text{BStep}(T, \mathfrak{a}_l, \alpha_l)$
4. *Full cycle enumerated?*
   if $\mathcal{O} = \mathfrak{a}_k$ for $k = 2l + 1$ or $k = 2(l + 1)$
   then return $\epsilon \leftarrow \alpha_k^{-1}$.

   repeat
5. *Baby step:*
   $(T, \mathfrak{a}_{2(d+L+1)}, \alpha_{2(d+L+1)}) = \text{BStep}(T, \mathfrak{a}_{2(d+L)}, \alpha_{2(d+L)})$
6. *Full cycle enumerated?*
   if $\mathcal{O} = \mathfrak{a}_k$ for $k = 2(d + L) + 1$ or $k = 2(d + L + 1)$
   then return $\epsilon \leftarrow \alpha_k$.
7. *Table look-up:*
   if there is a pair $(\mathfrak{a}_f, \alpha_f) \in T$ with $\mathfrak{b}_d = \mathfrak{a}_f$
   then return $\epsilon \leftarrow \alpha_f^{-1} \cdot \prod_{i=1}^{d} \beta_i$,
8. *Giant step:*
   $(\mathfrak{b}_{d+1}, \beta_{d+1}) = \text{GStep}(\mathfrak{b}_d, \mathfrak{a}_{2(d+L+1)}, \alpha_{2(d+L+1)})$.
9. $d \leftarrow d + 1$

PROOF Lemma 3.15 follows immediately from Lemma 2.19. ∎

Corollary 3.16 *Let* $L = \left\lceil \frac{1}{2} \log_2 \Delta \right\rceil$. TERRUNIT *computes sequences* (α_d) *and* (β_d) *so that* (3.1) *is satisfied for* $s_d = \operatorname{Log} \alpha_{2(d+L)}$ *and all* d. ∎

Proposition 3.17 *Given as input the discriminant* Δ *of a real quadratic order* $\mathcal{O}$, TERRUNIT

1. *executes no more than* $\frac{3}{2} \log_2 \Delta + \sqrt{2R/\log 2} + 1$ *calls to* BSTEP, *and no more than* $\log_2 \Delta + \sqrt{2R/\log 2}$ *calls to* GSTEP; *and*
2. *returns a power product representing the fundamental unit* $\epsilon > 1$.
3. *The run-time of* TERRUNIT *is bounded by* $O((\log \Delta + \sqrt{R}) \cdot (\log \Delta)^2)$.

PROOF *Claim 1:* TERRUNIT obviously terminates at the latest in step 6 after all reduced ideals in the principal cycle are enumerated.

If TERRUNIT does not already stop in step 4, then, by Corollary 3.16, it computes the baby-step sequence starting at $\mathcal{O}$, and a giant-step sequence governed by it. Hence Proposition 3.13 guarantees that it executes no more than $\lfloor E_{\mathrm{T}} \rfloor$ rounds of steps 5 through 9. Claim 1 follows.

Claim 2: If TERRUNIT terminates in steps 4 or 6 then the product returned equals the fundamental unit due to Proposition 2.13.

If, however, TERRUNIT terminates in step 8, then it has found the lexicographically smallest pair d, f with $0 < f \leq 2(d+1)$ and $\mathfrak{a}_f = \mathfrak{b}_d$. We have

$$(\prod_{i=1}^{e} \beta_i) = \mathfrak{b}_d = \mathfrak{a}_f = (\alpha_f)$$

so that ϵ defined by (3.4) is a unit. Applying Proposition 3.13 (2) shows $0 < \operatorname{Log} \epsilon^{-1} \leq R$. Hence ϵ generates the unit group, and $|\epsilon| > 1$. Since ϵ is a product of reducing numbers which are all positive, we see that indeed $\epsilon > 1$, and ϵ is fundamental.

Claim 3: The run-time bound follows from the bound on the number of calls to BSTEP and GSTEP from claim 1, and Lemmata 2.2 and 2.19. ∎

At the end of this section we give an example computation with TERRUNIT.

Example 3.18 Table 3.18 lists the ideals computed in the course of the execution of TERRUNIT for $\Delta = 2521$. Ideals in standard representation $a\mathbb{Z} + \mathbb{Z}\frac{b+\sqrt{\Delta}}{2}$ are listed as (a, b). Distances given are distances to $\mathcal{O}$.

Beyond the data shown in the table, TERRUNIT needs to store the generators of each $\mathfrak{b}_{d+1}$ relative to $\mathfrak{b}_d / \alpha_{2(d+1)}$ in order to compute the power product expansions for all β_j.

baby step ideals	distance	giant step ideals	distance
(30, 11)	2.203		
(20, 29)	2.426		
(21, 13)	3.085		
(28, 43)	3.350		
(6, 41)	4.630		
(35, 29)	5.776		
(12, 43)	6.435		
(14, 41)	7.715		
(15, 49)	8.861		
(2, 47)	11.065		
(39, 31)	12.770		
(10, 49)	13.491		
(3, 47)	15.694		
(26, 5)	17.400	(26,47)	-17.400
(24, 43)	17.500		
(7, 41)	18.779	(10,41)	-35.620
(30, 19)	19.926		
(18, 17)	20.324	(14,27)	-55.924
(31, 45)	20.676		
(4, 43)	22.129	(14,41)	-78.053
(42, 41)	23.409		
(5, 49)	24.555		

Tabelle 3.1: Baby and giant step ideals computed by TerrUnit for $\Delta = 2521$

The table shows that $\mathfrak{a}_8 = \mathfrak{b}_4$ (connected by an arrow). Using (3.4) this yields the fundamental unit. It is given by

$$\begin{aligned}\epsilon = {} & c(1,49)^5 \cdot c(30,11)^5 \cdot c(20,29)^5 \cdot c(21,13)^5 \cdot c(28,43)^5 \cdot c(6,41)^5 \\ & \cdot c(35,29)^5 \cdot c(12,43)^5 \cdot c(14,41)^4 \cdot c(15,49)^4 \cdot c(2,47)^4 \cdot c(39,31)^4 \\ & \cdot c(10,49)^4 \cdot c(3,47)^4 \cdot c(26,5)^3 \cdot c(24,43)^3 \cdot c(7,41)^2 \cdot c(30,19)^2 \\ & \cdot c(18,17) \cdot c(31,45) \cdot 26^{-1} \cdot c(26,99)^{-1} \cdot c(5,1)^{-1} \cdot 2^{-1}\end{aligned}$$

where $c(a,b) = (-b + \sqrt{\Delta})/(2a)$.

3.5 Discrete logarithm

In this section we show how to generalize TerrUnit to an algorithm for the computation of discrete logarithms in the group of invertible $\mathcal{O}$-ideals.

The proposed algorithm will be called TerrDL.

The problem it solves is an instance of the Extended Discrete Logarithm Problem with the groups $G = I_\Delta$ and $P = P_\Delta$. Given two invertible $\mathcal{O}$-ideals $\mathfrak{a}, \mathfrak{b}$ we are seeking to compute $n \in \mathbb{N}$, and $\gamma \in \mathcal{K}$ such that $\mathfrak{a} = \gamma \cdot \mathfrak{b}^n$ if they exist.

TerrDL is an instance of Algorithm 3.2 applied to the class group Cl_Δ of $\mathcal{O}$. In order to detect equality in the group $G = \mathrm{Cl}_\Delta$, we will use the results of Section 3.3, in particular a baby-step giant-step algorithm on the basis of Proposition 3.12.

Assume we are given two reduced $\mathcal{O}$-ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$. For $k \geq 1$, let $\mathfrak{a}_k, \mathfrak{b}_k$, and $\mathfrak{c}_k$ be defined recursively as follows:

$$\mathfrak{a}_0 = \mathcal{O}, \qquad \mathfrak{a}_k = \gamma_{a,k} \cdot \mathfrak{a}_{k-1}/\mathfrak{b} = \rho_0(\mathfrak{a}_{k-1}/\mathfrak{b}), \tag{3.6}$$

$$\mathfrak{b}_k = \gamma_{b,k} \cdot \mathfrak{c}_k/\mathfrak{a} = \rho_0(\mathfrak{c}_k/\mathfrak{a}), \tag{3.7}$$

$$\mathfrak{c}_0 = \mathcal{O}, \qquad \mathfrak{c}_k = \gamma_{c,k} \cdot \mathfrak{c}_{k-1}/\mathfrak{a}_k = \rho_0(\mathfrak{c}_{k-1}/\mathfrak{a}_k). \tag{3.8}$$

Obviously, $\mathfrak{a}_k, \mathfrak{b}_k \in \mathcal{R}_\Delta$, and $\mathfrak{a}_k \sim \mathfrak{b}^{-k}$, and $\mathfrak{b}_k \sim \mathfrak{b}^{k(k+1)/2}/\mathfrak{a}$. More precisely,

$$\mathfrak{a}_k = \prod_{i=1}^{k} \gamma_{a,i} \ \cdot \ \mathfrak{b}^{-k}, \tag{3.9}$$

$$\mathfrak{b}_k = \gamma_{b,k} \cdot \prod_{i=1}^{k} \gamma_{c,i}\gamma_{a,i}^{-k+i-1} \ \cdot \ \mathfrak{b}^{k(k+1)/2}/\mathfrak{a}, \tag{3.10}$$

$$\mathfrak{c}_k = \prod_{i=1}^{k} \gamma_{c,i}\gamma_{a,i}^{-k+i-1} \ \cdot \ \mathfrak{b}^{k(k+1)/2}. \tag{3.11}$$

TerrDL will compute baby-step sequences starting at $\mathfrak{a}_k$, and giant-step sequences starting at $\mathfrak{b}_k$ and $\mathfrak{c}_k$.

Once TerrDL finds a match, i.e. the current giant-step ideal occurs in the baby-step table, it has found indices $k \leq l$ such that $\mathfrak{a}_k \sim \mathfrak{b}_l$, or $\mathfrak{a}_k \sim \mathfrak{c}_l$, respectively. In the first case, $n = l(l+1)/2 + k$ is the sought discrete logarithm. In the second case n equals the order of $[\mathfrak{b}]$, and $[\mathfrak{a}] \notin \langle [\mathfrak{b}] \rangle$.

Once the discrete logarithm is found, it is easy to compute the generator of $\mathfrak{a}$ relative to $\mathfrak{b}^n$ on the basis of the data computed so far. TerrDL uses again a baby-step giant-step algorithm on the basis of Proposition 3.12.

Any sub-algorithm of TerrDL on the basis of Proposition 3.12 needs to make sure that

1. sufficiently many baby-step ideals are computed, say $2E_q$, from the baby-step sequence starting at $\mathfrak{a}_q$, and
2. some $\alpha_q \in \mathcal{K}$ are found such that (s_q) with $s_q = \operatorname{Log} \alpha_q$ can serve as spacing sequence for the giant-step sequences starting at $\mathfrak{b}_q$ and $\mathfrak{c}_q$ which are to be constructed.

The latter requirement is equivalent to (a) $s_q \leq r(\mathfrak{a}_p, 2E_p)$ for all $p \leq q$, and (b) $s_q > \frac{1}{2}\log\Delta + \sqrt{R\log 2}$.

We present two ways to arrive at appropriate values for E_q. Both approaches require pre-computation. The first approach is to execute TerrUnit, and compute a regulator approximation $R' > R$. Then set $E_q = \left\lceil \frac{1}{2}\log_2\Delta + \sqrt{R'/\log 2}\right\rceil$ for all q.

The second approach applies Proposition 3.13 (4). Compute a baby-step sequence and a giant-step sequence both starting at the same ideal, using $(s_{q,d})$ as the spacing sequence where $s_{q,2d} = \operatorname{Log}\alpha_{q,2(d+L+1)}$ and $\alpha_{q,d}$ is the sequence of generators of the baby-step sequence, $L = \left\lceil \frac{1}{2}\log_2\Delta\right\rceil$. Let e_q be the number of the round where baby-step and giant-step sequences meet. Then set $E_q = e_q + \left\lceil\log_2\Delta\right\rceil + 1$.

It depends on the relation between n and R which of the two approaches is faster for a given problem instance. The second approach involves exclusively operations with exact numbers, no approximations.

It is common to both pre-computations that they yield a baby-step table of reduced ideals on the principal cycle. α_q will be chosen from among the pre-computed generators of the ideals in this table by taking the generator of the ideal with an index close to the largest one for which 2(b) is satisfied.

We summarize the structure of TerrDL.

1. *Pre-computation* yields baby-step table S starting at $\mathcal{O}$, and, in approach 1, $E \approx \frac{1}{2}\log_2\Delta + \sqrt{R/\log 2}$.
2. Check principality of $\mathfrak{a}$ using S.
3. *Initialization:* $q \leftarrow 1, T \leftarrow S$.
4. **repeat**
5. Compute $\mathfrak{a}_q$. Store $\mathring{B}(\mathfrak{a}_q, E_q)$ in T. Use TerrUnit-like algorithm for enumeration, or set $E_q \leftarrow 2E$.
6. Compute $\mathfrak{c}_q$ and $\mathfrak{b}_q$.
7. Compute giant-step ideal sequence starting at $\mathfrak{b}_q$, and match against T. Any match yields n. In this case go to computation of relative generator.
8. Compute giant-step ideal sequence starting at $\mathfrak{c}_q$, and match against T. If a match is found, announce "$[\mathfrak{a}] \notin \langle[\mathfrak{b}]\rangle$", and exit.
9. $q \leftarrow q + 1$.
10. Compute generator of $\mathfrak{a}/\mathfrak{b}^n$ using S.

A detailed description of TerrDL can be found on page 43.

We will turn now to the description and analysis of the sub-algorithms of TerrDL: PrincipalBSTable for the pre-computation; BSteps for the enumeration of the baby-step ideal sequences; GSteps for the computation of the giant-step ideal sequences and their matching with the baby-step

table. GSteps will also be used to check whether a given ideal is principal, and, in the positive case, to determine its generator.

3.5.1 PrincipalBSTable1 and BSteps1

In this subsection we explain the baby-step algorithms specific to the first approach.

Lemma 3.19 *1. After termination of* PrincipalBSTable1 *we have* $0 < E - (\log_2 \Delta + \sqrt{R/\log 2}) < 4$, *and*

2. $S = \big((\mathfrak{a}_{0,d}, \alpha_{0,d})\big)_{d=1}^{F}$ *where* $\mathfrak{a}_{0,d} = (\alpha_{0,d}) = \rho^d(\mathcal{O})$ *and* F *is chosen such that* $R > r(\mathcal{O}, F) > \log \Delta + \sqrt{R \log 2}$ *or* $r(\mathcal{O}, F) = R$.

3. PrincipalBSTable1 *executes no more than* $\sqrt{2R/\log 2} + \log_2 \Delta + 2$ *times procedures* BStep, *and* GStep.

Proof *Claim 1:* According to Proposition 3.17, TerrUnit computes the fundamental unit ϵ. According to Lemma 2.23 Log returns an approximation R' of $R = \operatorname{Log} \epsilon$ with $R' \in \mathbb{Z}$ and $0 < R' - R < 2$. Thus we certainly have $E > \log_2 \Delta + \sqrt{R/\log 2}$ for E computed in step 2, and the other bound is easily obtained using $0 < R' - R < 2$ and $R \geq \log(1 + \sqrt{5}) - \log(2)$.

Claim 2 is immediate by construction. Indeed, for F returned by we know that $\mathfrak{a}_{0,F} = \mathcal{O}$, i.e. $r(\mathcal{O}, F) = R$, or $F > E > \log_2 \Delta + \sqrt{R/\log 2}$.

Claim 3: Proposition 3.17 also says that PrincipalBSTable1 executes no more than $N = \sqrt{2R/\log 2} + \log_2 \Delta + 1$ times procedures BStep and GStep in step 1. If PrincipalBSTable1 enters the while loop, then the total number of calls to BStep in step 1 and the while loop together does not exceed $M = \sqrt{(R+2)/\log 2} + \log_2 \Delta + 1$. The bound given in the lemma is always larger than both N (obviously) and M. The latter is clear for $R > 2$, and can easily be checked for the few discriminants yielding $R \leq 2$. The claim follows. ∎

Lemma 3.20 *Given an ideal* $\mathfrak{a} \in \mathcal{R}_\Delta$ *and* E *with* $0 \leq E - (\log_2 \Delta + \sqrt{E/\log 2}) \leq 4$, BSteps1

1. calls no more than $\sqrt{R/\log 2} + \log_2 \Delta + 4$ *times* BStep, *and*

2. returns $T = \big((\mathfrak{a}_d, \alpha_d)\big)_{d=1}^{F}$ *where* $\mathfrak{a}_d = \alpha_d \cdot \mathfrak{a} = \rho^d(\mathfrak{a})$ *and* F *is chosen such that* $R > r(\mathfrak{a}, F) > \log \Delta + \sqrt{R \log 2}$ *or* $r(\mathfrak{a}, F) = R$. ∎

3.5.2 PrincipalBSTable2 and BSteps2

In this subsection we explain the baby-step algorithms specific to the second approach. PrincipalBSTable2 and BSteps2 are essentially identical. We will call the common part BSSequence. It first constructs a baby-step and a giant-step sequence starting at the same ideal, and then expands the baby-step sequence once it has met the giant-step sequence by $2\lceil \log_2 \Delta \rceil + 1$ ideals.

Algorithm 3.5: Baby steps on the principal cycle

Description: Computes $E \approx \log_2 \Delta + \sqrt{R/\log 2}$, enumerates and stores ideals in $\mathring{B}(\mathcal{O}, F)$ such that $r(\mathcal{O}, F) \geq \max(R, \log \Delta + \sqrt{R \log 2})$

Input: Discriminant Δ

Output: $S = \mathring{B}(\mathcal{O}, F)$, F and E

PrincipalBSTable1(Δ)

1. $\epsilon \leftarrow$ TerrUnit(Δ). This also yields baby-step table $S = \mathring{B}(\mathcal{O}, F) = \big((\mathfrak{a}_{0,d}, \alpha_{0,d})\big)$.
2. $R' \leftarrow \text{Log}(\epsilon, 2)$. $E \leftarrow \lceil \log_2 \Delta + \sqrt{R'/\log 2} \rceil$.
3. **while** $F < 2E$
4. $\quad F \leftarrow F + 2$.
5. $\quad (S, \mathfrak{a}_{0,F}, \alpha_{0,F}) \leftarrow \text{BStep}(S, \mathfrak{a}_{0,F-2}, \alpha_{0,F-2})$.
6. $\quad$ **if** $\mathfrak{a}_{0,F} = \mathcal{O}$ **then break**.
7. $\quad$ **if** $\mathfrak{a}_{0,F} = \rho(\mathcal{O})$ **then** $S \leftarrow S \smallsetminus (\mathfrak{a}_{0,F}, \alpha_{0,F}), F \leftarrow F - 1$ and **break**.
8. **return** S, F, E.

Algorithm 3.6: Baby steps starting at $\mathfrak{a}$

Description: Enumerates and stores ideals in $B(\mathfrak{a}, 2\lceil E/2 \rceil)$

Input: Ideal $\mathfrak{a} \in \mathcal{R}_\Delta$, E

Output: Baby step table T, last computed ideal $\mathfrak{c}$

BSteps1($\mathfrak{a}, E$)

1. $T \leftarrow \emptyset, F \leftarrow 0, \mathfrak{a}_0 \leftarrow \mathfrak{a}$.
2. **while** $F < 2E$
3. $\quad F \leftarrow F + 2$.
4. $\quad (T, \mathfrak{a}_F, \alpha_F) \leftarrow \text{BStep}(T, \mathfrak{a}_{F-2}, \alpha_{F-2})$.
5. $\quad$ **if** $\mathfrak{a}_F = \mathfrak{a}$ **then break**.
6. $\quad$ **if** $\mathfrak{a}_F = \rho(\mathfrak{a})$ **then** $T \leftarrow T \smallsetminus (\mathfrak{a}_F, \alpha_F), F \leftarrow F - 1$ and **break**.
7. **return** $T, \mathfrak{a}_F$.

Algorithm 3.7: Baby step sequence construction (B)

Description: Computes E and enumerates ideals in $\mathring{B}(\mathfrak{a}, F)$ such that $r(\mathfrak{a}, F) > \log\Delta + \sqrt{2R\log 2}$ or $r(\mathfrak{a}, F) = R$.

Input: Ideal $\mathfrak{a} \in \mathcal{R}_\Delta$

Output: Baby step table $S = \mathring{B}(\mathfrak{a}, F)$, number F

BSSequence($\mathfrak{a}$)
1. $L \leftarrow \left\lceil \frac{1}{2}\log_2\Delta \right\rceil, S \leftarrow \emptyset, \mathfrak{a}_0 \leftarrow \mathfrak{a}, \alpha_0 \leftarrow 1,\ \mathfrak{b}_0 = \mathfrak{a}.$
2. **for** $d \leftarrow 1$ **to** L
3. $\quad (S, \mathfrak{a}_{2d}, \alpha_{2d}) \leftarrow \text{BStep}(S, \mathfrak{a}_{2(d-1)}, \alpha_{2(d-1)}),$
4. $\quad$ **if** there is a k such that $\mathfrak{a} = \mathfrak{a}_k \in S$ **then goto** step 15.
5. **repeat**
6. $\quad d \leftarrow d + 1.$
7. $\quad (S, \mathfrak{a}_{2d}, \alpha_{2d}) \leftarrow \text{BStep}(S, \mathfrak{a}_{2(d-1)}, \alpha_{2(d-1)}).$
8. $\quad$ **if** there is a pair $(\mathfrak{a}_k, \alpha_k) \in S$ with $\mathfrak{a} = \mathfrak{a}_k$ **then goto** 15.
9. $\quad$ **if** there is a pair $(\mathfrak{a}_k, \alpha_k) \in S$ with $\mathfrak{b}_{d-L-1} = \mathfrak{a}_k$ **then goto** 11.
10. $\quad (\mathfrak{b}_{d-L}, \beta_{d-L}) \leftarrow \text{GStep}(\mathfrak{b}_{d-L-1}, \mathfrak{a}_{2d}, \alpha_{2d}).$
11. **for** $i \leftarrow 1$ **to** $2L + 1$
12. $\quad d \leftarrow d + 1,$
13. $\quad (S, \mathfrak{a}_{2d}, \alpha_{2d}) \leftarrow \text{BStep}(S, \mathfrak{a}_{2(d-1)}, \alpha_{2(d-1)}),$
14. $\quad$ **if** there is a k such that $\mathfrak{a} = \mathfrak{a}_k \in S$ **then goto** step 15.
15. $F = 2d.$
16. **if** $\mathfrak{a}_{2d} = \mathfrak{a}$ **then** $S \leftarrow S \setminus \{(\mathfrak{a}_{2d}, \alpha_{2d})\}, F \leftarrow 2d - 1.$
17. **return** $S, F.$

The yield is a baby-step sequence with length sufficiently large to use it in a baby-step giant-step algorithm of classical type on this cycle without having access to a regulator estimate.

Lemma 3.21 *Given an ideal $\mathfrak{a} \in \mathcal{R}_\Delta$,* BSSequence

1. *executes no more than $\sqrt{2R/\log 2} + \frac{5}{2}\log_2\Delta + 4$ times procedure* BStep*, and no more than $\sqrt{2R/\log 2} + \log_2\Delta$ times procedure* GStep*;*
2. *returns $S = \big((\mathfrak{a}_d, \alpha_d)\big)_{d=1}^{F}$ where $\mathfrak{a}_d = \alpha_d \cdot \mathfrak{a} = \rho^d(\mathfrak{a})$ and F is chosen such that $R > r(\mathfrak{a}, F) > \log\Delta + \sqrt{2R\log 2}$ or $r(\mathfrak{a}, F) = R$.*

Proof *Claim 1:* Proposition 3.13 (1) guarantees that steps 5 through 10 are repeated no more than $\sqrt{2R/\log 2} + \log_2\Delta$ times. Adding the maximal number of calls to BStep in the loops starting in step 2 and step 11 we obtain the bound of the lemma.

Claim 2: If $\mathfrak{a}$ is ever encountered in steps 4, 8, or 14, then we obviously obtain S and F such that S contains all ideals on the cycle of $\mathfrak{a}$ together with their generators, and $r(\mathfrak{a}, F) = R$.

Assume this is not the case. Then Proposition 3.13 (4) shows that immediately before step 11 we have $r(\mathcal{O}, 2d)) > \sqrt{2R\log 2} - \frac{1}{2}\log 2$. Now the lower bound on $r(\mathfrak{a}, F)$ follows from the fact BSTEP increases the total length of the baby-step sequence at least by $\log 2$ (Lemma 2.8). ∎

Thus we let PRINCIPALBSTABLE2 be identical to a call to BSSEQUENCE with input $\mathcal{O}$, and BSTEPS2($\mathfrak{a}$) be identical to BSSEQUENCE($\mathfrak{a}$).

3.5.3 GSWIDTH

TERRDL uses GSWIDTH in order to obtain the width s of the giant-step sequences to be computed in every round, and obtain α such that $\operatorname{Log}\alpha = s$. This α is given by an index into the pre-computed baby-step sequence $\big((\mathfrak{a}_{0,d}, \alpha_{0,d})\big)$ on the principal cycle. In every round k, GSWIDTH computes b such that

$$\tfrac{1}{2}\log\Delta + \sqrt{R\log 2} < \operatorname{Log}\alpha_{0,b} < r(\mathfrak{a}_l, F_l), \quad \text{for all } l \le k, \tag{3.12}$$

where F_l is the number of elements of the baby-step sequence starting at $\mathfrak{a}_l$ that BSTEPS computed in round l. This index b is initially set to equal $F = F_0$, and then updated in each round by GSWIDTH.

Algorithm 3.8: Determination of giant-step width

Input: Baby step table $S = (\mathfrak{a}_d, \alpha_d)$ with $\mathfrak{a}_0 = \mathcal{O}$, principal ideal $\mathfrak{b} \in I_\Delta$, prior value of index b

Output: updated value of b

GSWIDTH($S, \mathfrak{b}, b$)

1. **if** there is a $d \in \mathbb{N}$ such that $(\mathfrak{a}_d, \gamma) = \text{CLOSEL}(\mathfrak{b})$ **and** $d < b$
2. **return** d.
3. **else return** b

Lemma 3.22 *Suppose we are given two baby-step sequences $S = \big((\mathfrak{a}_d, \alpha_d)\big)$ with $\mathfrak{a}_d = (\alpha_d) = \rho^d(\mathcal{O})$ and d running from 1 through some $F \in \mathbb{N}$, and $T = \big((\mathfrak{b}_d, \beta_d)\big)$ with $\mathfrak{b}_e = \beta_e \cdot \mathfrak{b} = \rho^e(\mathfrak{b})$ and e running from 1 through some $G \in \mathbb{N}$. Suppose we are given moreover an index $b \le F$ satisfying $R > r > c\log\Delta + \sqrt{R\log 2}$ or $r = R$ for $r = r(\mathcal{O}, b)$ with $c = 1/2$ and $r = r(\mathfrak{b}, G)$ with $c = 1$. Then we have for $b' = \text{GSWIDTH}(S, (\beta_G), b)$*

$$\tfrac{1}{2}\log\Delta + \sqrt{R\log 2} < \operatorname{Log}\alpha_{b'} \le \min(r(\mathcal{O}, b), r(\mathfrak{b}, G)), \quad \text{or} \quad \operatorname{Log}\alpha_{b'} = R \tag{3.13}$$

Algorithm 3.9: Computing giant-step ideals (A)

Input: Ideal $\mathfrak{b} \in \mathcal{R}_\Delta$, index b, baby-step table $T = \big((\mathfrak{a}_{k,f}, k)\big)$, integral $E > \log_2 \Delta + \sqrt{R/\log 2}$

Output: (k, α) such that $\mathfrak{b} = \alpha \cdot \mathfrak{a}_{k,0}$ or UNDEFINED

GSTEPS1$(\mathfrak{b}, b, T, E)$

1. **if** there is a pair $(\mathfrak{a}_{k,f}, k) \in T$ such that $\mathfrak{b} = \mathfrak{a}_{k,f}$ **then return** $k, \alpha_{k,f}$
2. $\mathfrak{b}_0 = \mathfrak{b}, \mathfrak{d} = (\alpha_{0,b})$.
3. **for** $e \leftarrow 1$ **to** E
4. $\quad(\mathfrak{b}_e, \beta_e) = \text{GSTEP}(\mathfrak{b}_{e-1}, \mathfrak{d}, \alpha_{0,b})$.
5. $\quad$**if** there is a pair $(\mathfrak{a}_{k,f}, k) \in T$ such that $\mathfrak{b}_e = \mathfrak{a}_{k,f}$ **then return** $k, \alpha \leftarrow \alpha_{k,f} \cdot \prod_{i=1}^{e} \beta_i^{-1}$
6. **return** UNDEFINED.

PROOF Keep the conditions of the lemma. Let $(\mathfrak{c}, \gamma) = \text{CLOSEL}((\beta_G))$. Then

$$\tfrac{1}{2} \log \Delta + \sqrt{R \log 2} < \operatorname{Log} \gamma\beta_G < r(\mathfrak{b}, G),$$

by Lemma 2.19. If there is an element in S of the form $((\gamma\beta_F), \alpha_d)$ for some $d < b$, then there is only one, and $b' = d$, and (3.13) is satisfied. Otherwise $b' = b$, and (3.13) holds by assumption. ∎

3.5.4 GSTEPS

Sub-algorithm GSTEPS is used to compute the giant-step ideal sequences in TERRDL, and detect a match between giant-step ideal sequence and baby-step table. As before we distinguish two approaches: (A) we have available some $E > \log_2 \Delta + \sqrt{R/\log 2}$, which was computed in PRINCIPALBSTABLE1; (B) we use only exact data and have to do without such E.

Lemma 3.23 *Given ideal $\mathfrak{b} \in \mathcal{R}_\Delta$; for all k in some finite set K containing 0 ideals $\mathfrak{a}_k \in \mathcal{R}_\Delta$ where $\mathfrak{a}_0 = \mathcal{O}$; baby-step table $T = \bigcup_{k \in K} \big((\mathfrak{a}_{k,d}, \alpha_{k,d}, k)\big)_{d=1}^{F_k}$ with $\mathfrak{a}_{k,d} = \alpha_{k,d} \cdot \mathfrak{a}_k = \rho^d(\mathfrak{a}_k)$; an index $b \le F_0$ satisfying $\operatorname{Log} \alpha_{0,b} \le \min(\, r(\mathfrak{a}_k, F_k) \mid k \in K\,)$ and $\frac{1}{2} \log \Delta + \sqrt{R \log 2} < \operatorname{Log} \alpha_{0,b} < R$ or $\operatorname{Log} \alpha_{0,b} = R$; and some E with $0 < E - (\log_2 \Delta + \sqrt{R/\log 2}) < 4$,* GSTEPS1

1. *executes no more than $\sqrt{R/\log 2} + \log_2 \Delta + 4$ calls to procedure* GSTEP*;*
2. *computes a giant-step sequence $\mathfrak{b}_d$ starting at $\mathfrak{b}$ that has spacing sequence $(\operatorname{Log} \alpha_{0,b})$;*
3. *returns a pair (l, α) for which $\mathfrak{b} = \alpha \cdot \mathfrak{a}_l$, unless the set $\{\, l \mid \mathfrak{b} \sim \mathfrak{a}_l \,\}$ is empty in which case it returns* UNDEFINED.

Algorithm 3.10: Computing giant-step ideals (B)

Description: Number of steps not known beforehand

Input: Ideal $\mathfrak{b} \in \mathcal{R}_\Delta$, current length index b, baby-step tables $S = \mathring{B}(\mathcal{O}, E) = \big((\mathfrak{a}_{0,d}, \alpha_{0,d}, 0)\big)$ and $T = \big((\mathfrak{a}_{k,f}, \alpha_{k,f}, k)\big)$

Output: k such that $\mathfrak{a}_k \sim \mathfrak{b}$ or UNDEFINED

GSTEPS2($\mathfrak{b}, b, S, T$)

1. **if** there is a pair $(\mathfrak{a}_{k,f}, k) \in T$ such that $\mathfrak{b} = \mathfrak{a}_{k,f}$ **then return** $k, \alpha \leftarrow \alpha_{k,f}$
2. $(U, G) = \big((\mathfrak{c}_d, \gamma_d)\big) \leftarrow$ BSSEQUENCE($\mathfrak{b}$).
3. $b' = \text{GSWIDTH}(S, (\gamma_G), b), \mathfrak{d} \leftarrow (\alpha_{0,b'})$.
4. **if** there is a pair $(\mathfrak{a}_{k,f}, k) \in T$ and $d \le G$ such that $\mathfrak{c}_d = \mathfrak{a}_{k,f}$ **then return** $k, \alpha \leftarrow \alpha_{k,f}\gamma_d^{-1}$
5. $\mathfrak{b}_0 = \mathfrak{b}, e \leftarrow 1$.
6. **repeat**
7. $\quad(\mathfrak{b}_e, \beta_e) = \text{GSTEP}(\mathfrak{b}_{e-1}, \mathfrak{d}, \alpha_{0,b'})$.
8. $\quad$**if** there is a pair $(\mathfrak{a}_{k,f}, k) \in T$ such that $\mathfrak{b}_e = \mathfrak{a}_{k,f}$ **then return** $k, \alpha \leftarrow \alpha_{k,f} \cdot \prod_{i=1}^{e} \beta_i^{-1}$
9. **until** there is a $d \le G$ such that $\mathfrak{b}_e = \mathfrak{c}_d$
10. **return** UNDEFINED.

PROOF Claim 1 is obvious. Claim 2 follows from Lemma 3.15.

If $\operatorname{Log} \alpha_{0,b} = R$, then $B(\mathfrak{a}_k, F_k) = C(\mathfrak{a}_k)$ for all k, and $\mathfrak{b}_e = \mathfrak{b}$ for all e. Thus $\mathfrak{b} \sim \mathfrak{a}_k$ if and only if there is an $f \le F_k$ such that $\mathfrak{a}_{k,f} = \mathfrak{b}$, which proves claim 3.

Assume $\operatorname{Log} \alpha_{0,b} \ne R$. Then $\frac{1}{2}\log\Delta + \sqrt{R \log 2} < \operatorname{Log} \alpha_{0,b} < R$. We apply Proposition 3.12. If $\mathfrak{a}_k \sim \mathfrak{b}$, then there are $f < F_k$, and $e < E_S < E$ such that $\mathfrak{a}_{k,f} = \mathfrak{b}_e$. GSTEPS1 detects this or an earlier match between the computed giant-step sequence and the union of baby-step sequences in T.

Vice versa, if GSTEPS1 finds a match $\mathfrak{a}_{k,f} = \mathfrak{b}_e$, then $\alpha_{k,f}\mathfrak{a}_k = \prod_{i=1}^{e} \beta_i \cdot \mathfrak{b}$, and, in particular, $\mathfrak{a}_k \sim \mathfrak{b}$. Thus $\mathfrak{b} = \alpha \cdot \mathfrak{a}_k$ with α as computed in step 1 or 5.

From the preceding it also follows that GSTEPS1 returns UNDEFINED if and only if $\{\, l \mid \mathfrak{b} \sim \mathfrak{a}_l \,\}$ is empty. This concludes the proof of claim 3. ∎

Without a lower bound for the number of giant steps to be computed available, we have to recognize when the giant-step sequence has "come full circle", i.e. has reached a total length exceeding R. To this end, we compute first a baby-step sequence of sufficient length with BSSEQUENCE.

Lemma 3.24 *Given an ideal $\mathfrak{b} \in \mathcal{R}_\Delta$; ideals $\mathfrak{a}_k \in \mathcal{R}_\Delta$ with $\mathfrak{a}_0 = \mathcal{O}$; baby-step tables $S = \big((\mathfrak{a}_{0,d}, \alpha_{0,d})\big)_{d=1}^{F}$ and $T = \bigcup_k \big((\mathfrak{a}_{k,d}, \alpha_{k,d}, k)\big)_{d=1}^{F_k}$ with $\mathfrak{a}_{k,d} =$*

$\alpha_{k,d} \cdot \mathfrak{a}_k = \rho^d(\mathfrak{a}_k)$*; and an index* $b \le F$ *satisfying* $\operatorname{Log}\alpha_{0,b} \le \min(\, r(\mathfrak{a}_k, F_k) \mid k\,)$ *and* $\frac{1}{2}\log\Delta + \sqrt{R\log 2} < \operatorname{Log}\alpha_{0,b} < R$ *or* $\operatorname{Log}\alpha_{0,b} = R$*,* GSteps2

1. *executes one call to* BSSequence*, one to* GSWidth*, and no more than* $E_{\mathrm{T}} = \log_2\Delta + \sqrt{R/\log 2}$ *calls to* GStep*;*
2. *computes a giant-step sequence* $\mathfrak{b}_d$ *starting at* $\mathfrak{b}$ *and with spacing sequence* $(\operatorname{Log}\delta)$*, where* δ *satisfies* $\operatorname{Log}\delta \le \min(\, r(\mathfrak{a}_k, F_k) \mid k\,)$ *and* $\frac{1}{2}\log\Delta + \sqrt{R\log 2} < \operatorname{Log}\delta < R$ *or* $\operatorname{Log}\delta = R$*;*
3. *returns* (k, α) *where* $\mathfrak{b} = \alpha \cdot \mathfrak{a}_k$*, unless the set* $\{\, l \mid \mathfrak{b} \sim \mathfrak{a}_l \,\}$ *is empty in which case it returns* Undefined.

Proof According to Lemma 3.21, we have $U = \big((\mathfrak{c}_d, \gamma_d)\big)_{d=1}^{G}$ with $R > r(\mathfrak{b}, G) > \log\Delta + \sqrt{2R\log 2}$ or $r(\mathfrak{b}, G) = R$ after step 2. From Lemma 3.22 we see that

$$\tfrac{1}{2}\log\Delta + \sqrt{R\log 2} < \operatorname{Log}\alpha_{b'} < \min_k(r(\mathfrak{b}, G), r(\mathfrak{a}_k, F_k)),$$

or $\operatorname{Log}\alpha_{b'} = R$ after step 3. Hence, claim 2 follows from Lemma 3.15.

This also shows that we can apply Proposition 3.12 to the baby-step sequence $(\mathfrak{c}_d)$ and the giant-step sequence $(\mathfrak{b}_e)$. Hence there are $d \le G$, and $e < E_{\mathrm{S}}$ such that $\mathfrak{b}_e = \mathfrak{c}_d$ which shows that GSteps2 necessarily terminates. Moreover, the number of calls to procedures BSSequence, GSWidth, and GStep are bounded as stated in claim 1.

We turn to the proof of claim 3. If GSteps2 returns k after having found a match between $\mathfrak{a}_{k,f} = \alpha_{k,f} \cdot \mathfrak{a}_k$ and $\mathfrak{b}_e = \prod \beta_i \cdot \mathfrak{b}$ or $\mathfrak{c}_d = \gamma_d \cdot \mathfrak{b}$, then $\mathfrak{a}_k \sim \mathfrak{b}$, and the field element α is indeed a generator of $\mathfrak{b}$ relative to $\mathfrak{a}_k$.

Suppose $\mathfrak{b} \sim \mathfrak{a}_k$ for some k, but GSteps2 returns Undefined. This assumption implies that GSteps2 does not find a match, i.e. never breaks execution neither in step 1, nor in step 4, nor in step 8.

Choose $\lambda \in \mathcal{K}$ such that $\mathfrak{b} = \lambda\mathfrak{a}_k$ and $0 < \operatorname{Log}\lambda \le R$. Applying Proposition 3.12 to $\mathfrak{a}_k$ and $(\mathfrak{b}_e)$ we find that there are $f \le F_k$ and $q < E_{\mathrm{S}}$ such that $\mathfrak{a}_{k,f} = \mathfrak{b}_q$. We choose the lexicographically smallest such pair (q, f). Let $\lambda_e = \lambda \cdot \prod_{i=1}^{e} \beta_i$ for all e. Then Proposition 3.12 also implies $\operatorname{Log}\lambda_q > 0$.

Let ϵ be the fundamental unit with $\operatorname{Log}\epsilon = R$. If $\operatorname{Log}(\epsilon/\lambda) + \operatorname{Log}\lambda_e \le \operatorname{Log}\gamma_G$, then also $\operatorname{Log}(\epsilon/\lambda) \le \operatorname{Log}\gamma_G$ and $\mathfrak{a}_k = (\epsilon/\lambda)\mathfrak{b}$ equals some $\gamma_g\mathfrak{b}$ with $g \le G$. Thus, GSteps2 detects a match already between the baby-step sequences in step 4, in contradiction to our assumption.

If $\operatorname{Log}(\epsilon/\lambda) + \operatorname{Log}\lambda_q > \operatorname{Log}\gamma_G$, then $\operatorname{Log}(\epsilon/\lambda) + \operatorname{Log}\lambda_p > \operatorname{Log}\gamma_G$ for all $p \le q$. Hence $\mathfrak{b}_p = (\epsilon/\lambda)\lambda_p\mathfrak{b}$ does not lie in $B(\mathfrak{b}, G)$ so that the loop in steps 6 through 9 reaches $e = q$. Thus GSteps2 detects the match in step 8, again in contradiction to our assumption.

The preceding also implies that $\{\, k \mid \mathfrak{b} \sim \mathfrak{a}_k \,\}$ is empty if and only if GSteps2 returns Undefined. Thus we have concluded the proof of claim 3.

■

3.5.5 Analysis of TerrDL

A listing of TerrDL can be found on page 43. The two variants of TerrDL involving other than exact data, or not, are obtained by substituting the correct variants of the sub-algorithms for PrincipalBSTable, BSteps, and GSteps. If variant A is used then it is understood that $E \approx \frac{1}{2}\log\Delta + \sqrt{R/\log 2}$ is computed in step 1, and implicitly used thereafter.

We convince ourselves that TerrDL produces correct output.

Proposition 3.25 *Given* $\mathfrak{a}, \mathfrak{b} \in I_\Delta$.

1. *Suppose* $[\mathfrak{a}] \in \langle[\mathfrak{b}]\rangle$. *Then* TerrDL$(\mathfrak{a}, \mathfrak{b})$ *terminates, and returns minimal* $n \in \mathbb{N}$, *and some* $\alpha \in \mathcal{K}$ *such that* $\mathfrak{a} = \alpha \cdot \mathfrak{b}^n$. *Otherwise* TerrDL$(\mathfrak{a}, \mathfrak{b})$ *announces "*$\mathfrak{a} \notin \langle[\mathfrak{b}]\rangle$*".*
2. *The run-time of* TerrDL *is bound by* $O(\sqrt{n}(\log\Delta + \sqrt{R\log 2}) \cdot (\log\Delta)^{(2+\varepsilon)})$, *(Variant A, the O-factor depending on* ε*), or* $O((\log\Delta + \sqrt{2R\log 2}) \cdot (\log\Delta)^2 \cdot \sqrt{n})$ *(Variant B)*

Proof The proposition follows from Proposition 3.6 once we have proven that equality in Cl_Δ is correctly tested by the matching of giant-step ideals with the baby-step table.

In step 1, PrincipalBSTable produces $S = \big((\mathfrak{a}_{0,d}, \alpha_{0,d})\big)_{d=1}^{F}$ with $\mathfrak{a}_{0,d} = (\alpha_{0,d}) = \rho^d(\mathcal{O})$ and F is chosen such that $R > r(\mathcal{O}, F) > \log\Delta + \sqrt{R\log 2}$ or $r(\mathcal{O}, F) = R$ according to Lemma 3.19 or Lemma 3.21, respectively. In variant A PrincipalBSTable also computes E such that $0 < E - (\log_2\Delta + \sqrt{R/\log 2}) < 3$. Thus we can apply Lemma 3.23 or Lemma 3.24 to the call of GSteps in step 4. Consequently, the check for $\mathfrak{a} \sim (1)$ is correctly performed, and in the positive case, a generator of $\mathfrak{b}_0$ found. This leads to correct output in step 5.

Likewise, $U_e = \big((\mathfrak{a}_{e,d}, \alpha_{e,d})\big)_{d=1}^{F_e}$ with $\mathfrak{a}_{e,d} = (\alpha_{e,d}) = \rho^d(\mathfrak{a}_e)$ and F_e is chosen such that $R > r(\mathfrak{a}_e, F_e) > \log\Delta + \sqrt{R\log 2}$ or $r(\mathfrak{a}_e, F_e) = R$. Now, Lemma 3.22 shows that GSWidth computes index b in step 10 such that we can apply Lemma 3.23 or Lemma 3.24 to the calls of GSteps in step 11 and step 14. Thus we are again assured that the test whether $\mathfrak{b}_e$ or $\mathfrak{c}_e$ are equivalent to some $\mathfrak{a}_d$ with $d \le e$ is properly performed.

Finally, TerrDL obtains a generator of $\mathfrak{a}$ relative to $\mathfrak{b}_n$ in steps 17 and 18 again due to Lemma 3.23 or Lemma 3.24 in combination with Lemma 2.19 which states the correctness of CloseL.

In order to obtain the run-time bound in claim 2, we collect the bounds from the preceding lemmata of this section. This is summarized in the table on the following page. ∎

Remark 1 It is possible to replace steps 17 and 18 with a recombination of the generator from intermediate factors with the help of formulae (3.6)

Algorithm 3.11: Deterministic Discrete logarithm algorithm in I_Δ

Description: Baby-step giant-step algorithm for the solution of the Extended Discrete Logarithm Problem in I_Δ.

Input: Discriminant Δ, $\mathcal{O}$-ideals $\mathfrak{a}, \mathfrak{b}$

Output: $n \in \mathbb{N}$ and $\alpha \in \mathcal{K}$ such that $n < \mathrm{ord}[\mathfrak{a}]$, and $\mathfrak{a} = \alpha\mathfrak{b}^n$

TerrDL$(\Delta, \mathfrak{a}, \mathfrak{b})$

Compute baby-step sequence on the principal cycle.

1. $(S, F) =$ PrincipalBSTable(Δ).
   This yields baby-step table $S = \big((\mathfrak{a}_{0,d}, \alpha_{0,d})\big)_{d=1}^{F}$.

Initialize baby-step table:

2. $T \leftarrow T_0 \leftarrow S \times \{0\}$.

Check principality of $\mathfrak{a}$.

3. $(\mathfrak{b}_0, \beta_0) =$ CloseL$(1/\mathfrak{a})$.
4. $(0, \alpha) =$ GSteps$(\mathfrak{b}_0, F, T_0)$.
5. **if** $\alpha \neq$ Undefined **then print** $(0, \beta_0/\alpha)$ and **exit**
6. $e \leftarrow 1, b \leftarrow F, \mathfrak{a}_0 \leftarrow \mathcal{O}, \mathfrak{c}_0 \leftarrow \mathcal{O}$.
7. **repeat**

   Baby steps:

8. $\mathfrak{a}_e \leftarrow \rho_0(\mathfrak{a}_{e-1}/\mathfrak{b}), (U_e, \mathfrak{f}_e) =$ BSteps$(\mathfrak{a}_e), T \leftarrow T \cup U_e \times \{e\}$.

   Giant steps:

9. $\mathfrak{c}_e \leftarrow \mathfrak{c}_{e,0} \leftarrow \rho_0(\mathfrak{c}_{e-1}/\mathfrak{a}_e)$, $\mathfrak{b}_e \leftarrow \mathfrak{b}_{e,0} \leftarrow \rho_0(\mathfrak{c}_{e,0}\mathfrak{b}_0)$.
10. $b \leftarrow$ GSWidth$(S, \mathfrak{f}_e/\mathfrak{a}_e, b)$.
11. $k =$ GSteps$(\mathfrak{b}_e, b, T_0, T)$.
12. **if** $k \neq$ Undefined
13. $n \leftarrow e(e+1)/2 + k$ and **goto** step 17.
14. $k =$ GSteps$(\mathfrak{c}_e, b, T_0, T)$.
15. **if** $k \neq$ Undefined **then print** "$\mathfrak{a} \notin \langle[\mathfrak{b}]\rangle$" and **exit**
16. $e \leftarrow e + 1$.

Compute generator of discrete logarithm relation.

17. $(\mathfrak{c}, \gamma) =$ CloseL$(\mathfrak{a}/\mathfrak{b}^n)$.
18. $(0, \alpha) =$ GSteps$(\mathfrak{c}, F, T_0)$.
19. **return** $(n, \alpha/\gamma)$.

---

through (3.8). This is considerably faster than execution of GSteps (affecting overall complexity by a small factor), but was skipped for simplicity of exposition.

Algorithm	Subalgorithm			Total Run Time Bound	Source
	Nr. of Calls	Name	Run Time Bound		
TerrDL				$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	Proposition 3.17
	1	PrincipalBSTable	$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	*or*	
	$\sqrt{2n}\ +2$	GSteps	$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^{2+\mathbf{e}})$	
	$\sqrt{2n}$	BSteps	$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$		
	$\sqrt{2n}$	GSWidth	$O((\log\Delta)^2)$		
	2	CloseL	$O((\log\Delta)^2)$		
PrincipalBSTable1				$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^{2+\mathbf{e}})$	Lemma 3.19 (3)
$\sqrt{R/\log 2}+\log_2\Delta+4$		BStep	$O((\log\Delta)^2)$		
	1	PPLog	$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^{2+\mathbf{e}})$		
BSSequence = PrincipalBSTable2 = BSteps2				$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	Lemma 3.21 (1)
$\sqrt{2R/\log 2}+\frac{5}{2}\log_2\Delta+4$		BStep	$O((\log\Delta)^2)$		
$\sqrt{2R/\log 2}+\log_2\Delta$		GStep	$O((\log\Delta)^2)$		
GSteps1				$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	Lemma 3.23 (1)
$\sqrt{R/\log 2}+\log_2\Delta+3$		GStep	$O((\log\Delta)^2)$		
GSteps2				$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	Lemma 3.24 (1)
	1	BSSequence	$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$		
	1	GSWidth	$O((\log\Delta)^2)$		
$\sqrt{R/\log 2}+\log_2\Delta$		GStep	$O((\log\Delta)^2)$		
BSteps1				$O((\sqrt{R}+\log\Delta+1)(\log\Delta)^2)$	Lemma 3.23 (1)
$\sqrt{R/\log 2}+\log_2\Delta+4$		BStep	$O((\log\Delta)^2)$		
GSWidth				$O((\log\Delta)^2)$	
	1	CloseL	$O((\log\Delta)^2)$		

Kapitel 4

Probabilistic Algorithms

In this chapter we will give probabilistic algorithms for the solution of the problems named in the introduction. All algorithms will execute using subexponentially bounded time and space. Proofs for run-time bounds and correctness of the algorithms will depend on some Generalized Riemann Hypothesis (GRH).

The chapter is structered as follows.

In the first section we will abstractly define relation lattices, and show that the solution of all problems from the introduction reduces to the computation of so-called "essentially full" relation lattices.

In the second section we will specialize to relation lattices on class group generators, and give the definitions for the objects (maps and lattices) used later on.

The third section recalls how the index calculus approach is applied in the setting of ideal (class) groups. The fourth section reviews the known result on the number of smooth reduced ideals in an order.

The fifth and sixth sections deal with the random choice of reduced ideals. The fifth section deals with the random choice of an ideal class. The sixth one explains for the case of positive discriminants how to choose a random reduced ideal in a cycle of ideals. The algorithms presented choose reduced ideals in a nearly uniform manner. In the real quadratic case, the deviation from uniformness is bounded by a small power of the logarithm of the discriminant.

In the seventh section we introduce the algorithms for the generation of "potential" relations. A potential relation is a pair or triple of objects from which a relation can be computed if the reduced ideal contained is smooth. In the following we will test smoothness with Bernstein's algorithm for the factorization of large sets of integers, applying it, in consequence, to large sets of potential relations.

The eighth section explains Seysen's method for generating a full rank relation lattice. The basic change here is the use of Bernstein's algorithm

for factoring and the extension to the case of positive discriminants.

The ninth and tenth sections contain the essentially new results of this chapter.

In the ninth section we will show how to compute a set of relations that generate an essentially full relation lattice. This can be done an order of magnitude faster, than, e.g., in Hafner and McCurley's algorithm since it will be shown that it suffices to compute relation vectors that have only $O^{\sim}(1)$ non-zero entries.

In the tenth section the regulator algorithm is presented. The computation of the regulator is not only of independent interest. It also allows us to check the whether the relation lattice computed is indeed essentially full, or not.

In the eleventh section we recall how this check is performed on the basis of the analytic class number formula. We conclude the chapter with the twelfth section where we summarize the results in two theorems stating the existence of algorithms for GOP, GSP, (E)DLP in time $L_{|\Delta|}(\frac{1}{2}, 3/\sqrt{8})$.

4.1 Abstract relation lattices

Let G be a commutative group, written multiplicatively. Assume that $\mathcal{G} = (g_1, \ldots, g_n)$ is an *ordered* generating set of G. Let $n = \operatorname{card} \mathcal{G}$. We have the following obvious map induced by $\mathcal{G}$ and the $\mathbb{Z}$-action on G.

$$\phi_{\mathcal{G}} : \mathbb{Z}^n \longrightarrow G : (a_1, \ldots, a_n) \longmapsto \prod_{i=1}^{n} g_i^{a_i} \tag{4.1}$$

The kernel $\mathcal{L} = \mathcal{L}_{\mathcal{G}}$ of $\phi_{\mathcal{G}}$ is a sub-lattice of $\mathbb{Z}^n$. It is called the *relation lattice* corresponding to $\mathcal{G}$. Individual elements of this lattice are called *relations*. If G is finite, then $\mathcal{L}$ has full rank, and the determinant of $\mathcal{L}$ equals the cardinality of G. In all what follows we will presuppose throughout that G is finite.

Assume we are given an ordered set $\mathcal{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_m)$ of relations $\mathcal{L}$. We assign a matrix $A = A(\mathcal{V})$ to $\mathcal{V}$ which contains the vectors $\mathbf{v}_i$ as column vectors. The column vectors of all matrices obtained from $A(\mathcal{V})$ by multiplication with another matrix from the right are obviously also elements of $\mathcal{L}$.

Let $\mathcal{V}$ be some finite sub-set of $\mathcal{L}$, and $\mathcal{M}$ the sub-lattice of $\mathcal{L}$ generated by $\mathcal{V}$. Compute the HNF of $A(\mathcal{V})$. Let H be its principal $n \times n$ minor. Since there is exactly one column-reduced matrix H whose columns span a given lattice, we see that H is an invariant of $\mathcal{M}$. Hence we will denote it by $H(\mathcal{M})$. In this sub-section, we will also call H the HNF of $A(\mathcal{V})$. In Chapter 5 we will discuss the computation of the HNF of matrices.

Sub-lattices, sub-groups. If $\mathcal{F} = (g_1, \ldots, g_k, f_{k+1}, \ldots, f_n)$ is a generating set of Abelian group F and G is the sub-group of F generated by

$\mathcal{G} = (g_1, \ldots, g_k)$, then we have the following commutative diagram.

$$\begin{array}{ccccccccc}
 & & 0 & & 0 & & 1 & & \\
 & & \downarrow & & \downarrow & & \downarrow & & \\
0 & \longrightarrow & \mathcal{L}_{\mathcal{G}} & \longrightarrow & \mathbb{Z}^k & \xrightarrow{\phi_{\mathcal{G}}} & G & \longrightarrow & 1 \\
 & & \downarrow & & \downarrow{\scriptstyle \iota} & & \downarrow & & \\
0 & \longrightarrow & \mathcal{L}_{\mathcal{F}} & \longrightarrow & \mathbb{Z}^n & \xrightarrow{\phi_{\mathcal{F}}} & F & \longrightarrow & 1 \\
 & & \downarrow & & \downarrow{\scriptstyle \pi} & & \downarrow & & \\
0 & \longrightarrow & \mathcal{L}_{\mathcal{F} \setminus \mathcal{G}} & \longrightarrow & \mathbb{Z}^{n-k} & \xrightarrow{\phi_{\mathcal{F} \setminus \mathcal{G}}} & F/G & \longrightarrow & 1 \\
 & & \downarrow & & \downarrow & & \downarrow & & \\
 & & 0 & & 0 & & 1 & &
\end{array} \tag{4.2}$$

where

$$\iota : \mathbb{Z}^k \longrightarrow \mathbb{Z}^n \, : \, (v_1, \ldots, v_k) \longmapsto (v_1, \ldots, v_k, 0, \ldots, 0), \tag{4.3}$$

and

$$\pi : \mathbb{Z}^n \longrightarrow \mathbb{Z}^{n-k} \, : \, (v_1, \ldots, v_n) \longmapsto (v_{k+1}, \ldots, v_n). \tag{4.4}$$

Lemma 4.1 *If G equals F then for any $v \in \mathbb{Z}^n$ there exists a $v' \in \mathbb{Z}^k$ such that $v \equiv \iota(v') \mod \mathcal{L}_{\mathcal{F}}$ and $H(\mathcal{L}_{\mathcal{F}})$ has the form*

$$\begin{pmatrix} H(\mathcal{L}_{\mathcal{G}}) & * \\ 0 & I_{(n-k)\times(n-k)} \end{pmatrix} \tag{4.5}$$

Proof The first claim can be seen by an easy diagram chase. We turn to the second claim of the lemma.

Since $\mathcal{G}$ generates F, we find for any $j > k$ exponents ν_{ij} with $i = 1, \ldots, k$, such that

$$f_j = \prod_{i=1}^{k} g_i^{\nu_{ij}}.$$

Let $\mathbf{e}_r$ denote the r-th column of the $n \times n$ identity matrix. Then we have seen that $\mathbf{e}_j \equiv \sum_{i=1}^{k} \nu_{ij} \cdot \mathbf{e}_i \mod \mathcal{L}_{\mathcal{F}}$ for all $j \geq k+1$ which implies that the diagonal entries of the lower right $(n-k) \times (n-k)$ minor of $H = H(\mathcal{L}_{\mathcal{F}})$ are all 1. Hence H has the form claimed by the lemma. ∎

Lemma 4.2 *Let $\mathcal{M}$ be a full-rank sub-lattice of $\mathcal{L}_{\mathcal{F}}$. Then $H(\pi(\mathcal{M}))$ is the lower-right principal $(n-k) \times (n-k)$ minor of $H(\mathcal{M})$.*

PROOF Let $\mathcal{V}$ be such that $A(\mathcal{V}) = H(\mathcal{M})$. Set $\mathcal{W} = \pi(\mathcal{V}) \setminus \{0\}$. Then $A(\mathcal{W})$ is the lower-right principal $(n-k) \times (n-k)$ minor of $H(\mathcal{M})$, and, hence, column-reduced. Since $\mathcal{W}$ generates $\pi(\mathcal{M})$ it follows that $H(\pi(\mathcal{M})) = A(\mathcal{W})$. ∎

Lemma 4.3 *Let $\mathcal{M}$ be a sub-lattice of $\mathcal{L}_\mathcal{F}$ with $\iota(\mathcal{L}_\mathcal{G}) \subset \mathcal{M}$. Then $H(\mathcal{L}_\mathcal{G})$ is the upper-left principal $k \times k$ minor of $H(\mathcal{M})$.*

PROOF Let $\mathcal{W}$ be such that $A(\mathcal{W}) = H(\mathcal{M})$. Let $\mathcal{V} = \mathcal{W} \cap \operatorname{Im} \iota$. Then $A(\iota^{-1}(\mathcal{V}))$ is the upper-left principal $k \times k$ minor of $H(\mathcal{M})$, and, hence, column-reduced. We show that $\iota^{-1}(\mathcal{V})$ generates $\mathcal{L}_\mathcal{G}$.

Let $\mathbf{v} \in \mathcal{L}_\mathcal{G}$. Then $\iota(\mathbf{v}) = \sum_{\mathbf{w} \in \mathcal{W}} a_\mathbf{w} \mathbf{w}$. Since $A(\mathcal{W})$ is column-reduced, we have $a_\mathbf{w} = 0$ if $\mathbf{w} \notin \operatorname{Im} \iota$. Thus $\mathbf{v}$ lies in the span of $\iota^{-1}(\mathcal{V})$. ∎

This lemma prompts the following definition:

Definition 4.4 Let $\mathcal{G} \subset \mathcal{F}$ be two finite generating sets for the abelian group G. A sub-lattice $\mathcal{M} \subset \mathcal{L}_\mathcal{F}$ is called *essentially full* with respect to $\mathcal{G}$ if $\iota(\mathcal{L}_\mathcal{G}) \subset \mathcal{M}$.

If $\mathcal{M}$ is an essentially full sub-lattice of $\mathcal{L}_\mathcal{F}$ with respect to $\mathcal{G} \subset \mathcal{F}$, then $H(\mathcal{M})$ has the following form:

$$\begin{pmatrix} H(\mathcal{L}_\mathcal{G}) & * \\ 0 & * \end{pmatrix}$$

Algorithms EssHNF and EssDet which are explained in Chapter 5 compute $H(\mathcal{L}_\mathcal{G})$ and its determinant from a generating set $\mathcal{V}$ of $\mathcal{M}$ with cardinality m in time $O^\sim(m^3 \log \|A(\mathcal{V})\|)$.

In the following we will list a number of basic facts linking $H(\mathcal{L}_\mathcal{F})$ or, more generally, $H(\mathcal{M})$ for essentially full sub-lattices $\mathcal{M}$ to data about G.

Lemma 4.5 (Group order) *Let $\mathcal{G}$ be a generating set for the commutative group G. Then*

$$\det H(\mathcal{L}_\mathcal{G}) = \det \mathcal{L}_\mathcal{G} = \operatorname{card} G.$$

PROOF This follows from $\mathbb{Z}^n / \mathcal{L}_\mathcal{G} \simeq G$. ∎

Lemma 4.6 (Element order)
1. *Assume the group G is cyclic. Let g be a generator. If $(v) = H(\mathcal{L}_{\{g\}})$, then $\operatorname{ord} g = v$.*
2. *Let $\mathcal{G} = (g = g_1, \ldots, g_n)$ be a generating set for G, and $\mathcal{M}$ an essentially full sub-lattice of $\mathcal{L}_\mathcal{G}$ with respect to $\{g\}$. If $H(\mathcal{M}) = (h_{ij})$, then $\operatorname{ord} g = h_{11}$.*

PROOF Consider the first statement. We know $g^v = 1$ since (v) is a relation, and $g^w = 1$ implies $v \mid w$ since (v) generates $\mathcal{L}_{\{g\}}$.

The second statement follows from the first and Lemma 4.3. ∎

Lemma 4.7 (Order of element modulo subgroup)
Let $\mathcal{F} = (g_1, \ldots, g_k, f = f_{k+1}, f_{k+2}, f_n)$ be a generating set for the commutative group F. Let G be the sub-group of F generated by $\mathcal{G} = (g_1, \ldots, g_k)$. If $\mathcal{M}$ is a sub-lattice of $\mathcal{L}_{\mathcal{F}}$ essentially full with respect to $\mathcal{G} \cup \{f\}$, and $H(\mathcal{M}) = (h_{ij})$, then $\operatorname{ord}(f \bmod G) = h_{(k+1),(k+1)}$.

Proof Apply Lemma 4.3 to $\mathcal{G} \cup \{f\}$ and $\mathcal{F}$, then Lemma 4.2 to $\mathcal{G}$ and $\mathcal{G} \cup \{f\}$ to see that $(h_{(k+1),(k+1)}) = \mathcal{L}_{f \bmod \mathcal{G}}$. Hence $\operatorname{ord} f \bmod \mathcal{G} = h_{(k+1),(k+1)}$ by Lemma 4.6. ∎

Corollary 4.8 (Discrete logarithm) *Let G be a commutative group. Fix $g, h \in G$. Assume G is generated by the set $\mathcal{G} = (g, h, g_3, \ldots, g_n)$. Let $\mathcal{M}$ be a sub-lattice of $\mathcal{G}$ essentially full with respect to $\{g, h\}$, and $H(\mathcal{M}) = (h_{ij})$. If $h_{2,2} \neq 1$, then $h \notin \langle g \rangle$. If on the other hand, $h_{2,2} = 1$, then the discrete logarithm of h with respect to g is $h_{1,1} - h_{1,2}$.*

Proof Keep the set-up of the corollary. Then $h_{1,1} = \operatorname{ord} g$, and $h_{22} = \operatorname{ord}(h \bmod \langle g \rangle)$ by Lemma 4.7. If and only if $h_{22} > 1$, then $h \notin \langle g \rangle$, and the discrete logarithm is undefined. Assume $h_{22} = 1$. The second column of $H(\mathcal{M})$ represents an element of $\mathcal{L}_{\mathcal{G}}$, i.e. $g^{h_{12}} \cdot h^{h_{22}} = 1$. Hence $g^{h_{1,1} - h_{1,2}} = h$ and $0 \leq h_{1,1} - h_{1,2} < h_{1,1} = \operatorname{ord} g$. ∎

Thus, we have reduced all of the previous problems to the generation of a set of relations that generate an essentially full relation lattice.

4.2 Relation lattices on class group generators

In this and the following sections we will return to the special case that G is the class group of a maximal quadratic order.

Thus we let $\mathcal{O}$ be a maximal quadratic order, and $\mathcal{K}$ its fractional field which we assume to be embedded into $\mathbb{R}$, or $\mathbb{C}$, depending on whether $\mathcal{O}$ is real or imaginary. We denote the discriminant of $\mathcal{O}$ by Δ, the group of invertible $\mathcal{O}$-ideals by I_Δ, its subgroup of principal ideals by P_Δ, the class group I_Δ / P_Δ of $\mathcal{O}$ by Cl_Δ, the class number by h_Δ, and the non-trivial automorphism of $\mathcal{K}$ by σ. The ideal class represented by an ideal $\mathfrak{a}$ will be denoted by $\bar{\mathfrak{a}}$. In the real quadratic case we will denote the fundamental unit by ϵ_Δ and the regulator by R_Δ.

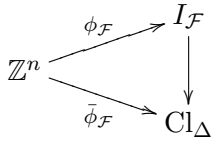
In order to utilize the set-up from the previous section we need a generating set for the class group of $\mathcal{O}$. For this we rely on the following well-known result by Bach [Bac90].

Proposition 4.9 (GRH) *For all fundamental discriminants Δ with $|\Delta| \gg 0$, the set $\mathcal{G} = \{ \mathfrak{p} \in I_\Delta \text{ prime} \mid \mathrm{N}\mathfrak{p} < 3 \log^2 |\Delta| \}$ generates the class group of the order with discriminant Δ.*

For the purposes of discrete logarithm computation we will later enlarge $\mathcal{G}$ so that it contains base and argument of the discrete logarithm problem to be solved.

In this section we let $\mathcal{F} \subset \mathcal{I}_\Delta$ denote some large superset of $\mathcal{G}$. The cardinality of $\mathcal{F}$ will be denoted by n, the free sub-group of I_Δ generated by the elements of $\mathcal{F}$ by $I_\mathcal{F}$. We assume that $\mathcal{F}$ is ordered in such a way that any element of $\mathcal{G}$ is smaller than any element of $\mathcal{F} \setminus \mathcal{G}$.

As in (4.1), given the ordered set $\mathcal{F}$ we have maps $\phi_\mathcal{F}$ and $\bar{\phi}_\mathcal{F}$



Due to unique factorization of ideals in the Dedekind domain $\mathcal{O}$, the map $\phi_\mathcal{F}$ is an isomorphism. Since the classes represented by elements of $\mathcal{F}$ generate Cl_Δ, the map $I_\mathcal{F} \longrightarrow \mathrm{Cl}_\Delta$ is surjective, and hence so is $\bar{\phi}_\mathcal{F}$. Thus we can directly apply the framework of the preceding section. In doing so, we will call sub-lattices of $\mathcal{L}_\mathcal{F}$ essentially full if they are essentially full with respect to $\mathcal{G}$.

In the imaginary quadratic case, Hafner and McCurley [HM89] have shown how to construct a generating set $\mathcal{V}$ for $\mathcal{L}_\mathcal{F}$ for sufficiently large $\mathcal{F}$. We will improve upon their algorithm by showing how to find a generating set of an essentially full sub-lattice of $\mathcal{L}_\mathcal{F}$ in substantially less time.

In the real quadratic case, we are not only interested in the class group, but also in generators of reduced principal ideals. The computation of the generator of a reduced principal ideal is polynomial time equivalent to the computation of its logarithm. Thus we extend our framework by a real component.

The idea of this extension was first presented in [Buc90]. There, Buchmann generalized Hafner and McCurley's idea and introduced, for arbitrary maximal orders of global number fields with discriminant Δ, lattices $\tilde{\mathcal{L}}_\mathcal{F} \subset \mathbb{Z}^m \oplus \mathbb{R}$ with determinant $h_\Delta R_\Delta$.

We recall the definition of $\tilde{\mathcal{L}}_\mathcal{F}$ in the real quadratic case. Roughly spoken, it extends the lattice of relations over $\mathcal{F}$ to contain pairs $(\mathbf{v}, \log|\alpha|)$ where α generates the principal ideal corresponding to $\mathbf{v}$.

Let

$$\theta : \mathcal{K}^* \longrightarrow P_\Delta : \alpha \longmapsto (\alpha),$$

and $O_\mathcal{F} = \theta^{-1}(I_\mathcal{F} \cap P_\Delta)$.

We define the lattice $\tilde{\mathcal{L}}_\mathcal{F}$ to be the image of $O_\mathcal{F}$ under $(\phi_\mathcal{F}^{-1} \circ \theta, \mathrm{Log})$. We will call its elements *extended relations*. The positive pre-image of an extended relation under $(\phi_\mathcal{F}^{-1} \circ \theta, \mathrm{Log})$ is called its generator. For any $\tilde{\mathbf{v}} =$

$(\mathbf{v}, \operatorname{Log}\alpha) \in \tilde{\mathcal{L}}_{\mathcal{F}}$, we call $\mathbf{v} = \pi(\tilde{\mathbf{v}})$ its integral part. For sub-lattices $\tilde{\mathcal{M}}$ of $\tilde{\mathcal{L}}_{\mathcal{F}}$, we will denote $\pi(\tilde{\mathcal{M}})$ also simply by $\mathcal{M}$.

From the diagram

$$\begin{array}{ccccccccc} 1 & \longrightarrow & \pm 1 & \longrightarrow & \mathcal{O}_{\mathcal{F}} & \xrightarrow{(\phi_{\mathcal{F}}^{-1}\circ\theta,\operatorname{Log})} & \mathbb{Z}^m \oplus \mathbb{R} & & \\ & & \downarrow & & \| & & \pi\downarrow & & \\ 1 & \longrightarrow & \mathcal{O}^* & \longrightarrow & \mathcal{O}_{\mathcal{F}} & \xrightarrow{\phi_{\mathcal{F}}^{-1}\circ\theta} & \mathbb{Z}^m & \xrightarrow{\bar{\phi}_{\mathcal{F}}} \operatorname{Cl}_\Delta & \longrightarrow & 1 \end{array}$$

we see that $\pi|\tilde{\mathcal{L}}_{\mathcal{F}}$ has kernel $(0, R_\Delta\mathbb{Z})$ and, using e.g. the snake lemma, that the sequence

$$0 \longrightarrow \mathbb{R}/R_\Delta\mathbb{Z} \longrightarrow (\mathbb{Z}^m \oplus \mathbb{R})/\tilde{\mathcal{L}}_{\mathcal{F}} \longrightarrow \operatorname{Cl}_\Delta \longrightarrow 1$$

is exact. The last exact sequence is compatible with the canonical measures on the groups which implies, as can also be seen directly, that

$$\det \tilde{\mathcal{L}}_{\mathcal{F}} = h_\Delta \cdot R_\Delta. \tag{4.6}$$

Also in [Buc90], Buchmann showed how to produce a generating set for $\tilde{\mathcal{L}}_{\mathcal{F}}$. We will slightly modify his approach. We will show how to obtain

1. a generating set $\mathcal{V}$ for an essentially full sub-lattice of $\mathcal{L}_{\mathcal{F}}$, and
2. a generating set $\mathcal{E}$ for $\ker \pi|\tilde{\mathcal{L}}_{\mathcal{F}}$.

The order of a class in the class group, the exponent of a discrete logarithm relation in the class group and the structure of the latter is obtained from $\mathcal{V}$ using the framework of the preceding section.

Computing the real GCD of $\mathcal{E}$ yields the regulator R_Δ. An algorithm for the computation of real GCDs can be found in [Mau00].

Finally, we obtain a generator of a given reduced principal ideal $\mathfrak{a}$ by adding it to $\mathcal{F}$, and finding some element $v = (\mathbf{v}, \delta)$ in $\tilde{\mathcal{L}}_{\mathcal{F}}$ such that $\mathfrak{a} = \phi_{\mathcal{F}}(\mathbf{v})$. Subtracting an appropriate multiple of R_Δ from δ, and applying the methods of [BTW95] we arrive at a generator of $\mathfrak{a}$ in compact representation.

4.3 Random relations

In this section we will show how to obtain random relations. This is done using the classical index calculus method.

Imaginary quadratic case. Given a large generating set $\mathcal{F}$ for the class group Cl_Δ with cardinality n we will seek to obtain pairs of elements $\mathbf{v}_1, \mathbf{v}_2$ in $\mathbb{Z}^n$ with the same image under $\bar{\phi}_{\mathcal{F}}$. This is done in the following steps:

1. Choose a random vector $\mathbf{v}_1$.
2. Find a reduced ideal $\mathfrak{a}$ in the class $\bar{\phi}_{\mathcal{F}}(\mathbf{v}_1)$.

3. Try to factor $\mathfrak{a}$ over $\mathcal{F}$; if successful, this yields $\mathbf{v}_2$ with $\phi_{\mathcal{F}}(\mathbf{v}_2) = \mathfrak{a}$.

We clearly have $\mathbf{v}_1 - \mathbf{v}_2 \in \mathcal{L}_{\mathcal{F}}$.

Real quadratic case. Given a large generating set $\mathcal{F}$ for the class group Cl_Δ with cardinality n we will seek to obtain pairs of elements $\mathbf{v}_1, \mathbf{v}_2$ in $\mathbb{Z}^n$ and $\alpha \in \mathcal{K}$ with $\phi_{\mathcal{F}}(\mathbf{v}_1) = \alpha \cdot \phi_{\mathcal{F}}(\mathbf{v}_2)$. This is done in the following steps:

1. Choose a random vector $\mathbf{v}_1$.
2. Choose randomly a reduced ideal $\mathfrak{a}$ in the class $\bar{\phi}_{\mathcal{F}}(\mathbf{v}_1)$, and α such that $\mathfrak{a} = \alpha \cdot \phi_{\mathcal{F}}(\mathbf{v}_1)$.
3. Try to factor $\mathfrak{a}$ over $\mathcal{F}$; if successful, this yields $\mathbf{v}_2$ with $\phi_{\mathcal{F}}(\mathbf{v}_2) = \mathfrak{a}$.

We then clearly have $(\mathbf{v}_1 - \mathbf{v}_2, \operatorname{Log} \alpha) \in \tilde{\mathcal{L}}_{\mathcal{F}}$.

In the following sections we will show how to choose $\mathbf{v}$, and, in the real quadratic situation, $\mathfrak{a}$ such that the probability the described procedures succeed, i.e. the chosen reduced ideals factor over given factor base $\mathcal{F}$, is sufficiently large.

4.4 Smooth reduced ideals

In this section, we establish a lower bound for the number of those reduced ideals in $\mathcal{O}$ that factor over a given factor base $\mathcal{F}$. For this we rely on results by Seysen [Sey87].

As factor base we choose the set of all prime ideals above prime numbers splitting in $\mathcal{K}/\mathbb{Q}$ whose norm is smaller then a given bound $y \in \mathbb{R}$. We will denote the set of such prime numbers by $\mathcal{P}_y$,

$$\mathcal{P}_y = \{\, p \text{ prime} \mid \left(\tfrac{\Delta}{p}\right) = 1, p < y \,\}$$

and the factor base by $\mathcal{F}_y$,

$$\mathcal{F}_y = \{\, \mathfrak{p} \in I_\Delta \mid \mathrm{N}\,\mathfrak{p} \in \mathcal{P}_y \,\}.$$

For both imaginary and real quadratic $\mathcal{O}$, we know that primitive ideals with sufficiently small norm are reduced.

Lemma 4.10 *A primitive $\mathcal{O}$-ideal with norm smaller $\sqrt{|\Delta|}/2$ is reduced.*

Proof This is well known (since Gauss). For proofs in the language of ideals, see, e.g., [BW88] for the case $\Delta < 0$, and [Wil85] for $\Delta > 0$. ∎

Hence it suffices to bound from below the number of all integers which factor into primes in $\mathcal{P}_y$. We define the counting function

$$\psi_\Delta(x, y) = \operatorname{card}\{\, a \in \mathbb{N} \mid a < x; \text{ and } p \in \mathcal{P}_y \text{ for all primes } p \text{ dividing } a \,\}.$$

In [Sey87], Seysen proved a lower bound for ψ_Δ.

Proposition 4.11 (ERH) *For any $\epsilon > 0$ there is a $c_1(\epsilon)$ such that for any x, y, and Δ satisfying*

$$\max\big((\log x)^{(1+\epsilon)}, (\log|\Delta|)^{(2+\epsilon)}\big) \le y \le \exp\big((\log x)^{(1-\epsilon)}\big)$$

the following property holds:

$$\psi_\Delta(x,y) \ge x \cdot \exp\big(-u \cdot (\log u + \log\log u + c_1(\epsilon))\big)$$

where $u = (\log x)/(\log y)$.

Note that Seysen stated this theorem only for $\Delta < 0$. The proof he gives, however, is independent of the sign of Δ. The theorem also follows from the results of [BH96].

Corollary 4.12 (ERH) *For any $z > 0$, there exist $\Delta_0(z)$ and $c_2(z) > 0$ such that for $|\Delta| > \Delta_0(z)$, and smoothness bound $y = L_{|\Delta|}(\frac{1}{2}, z)$, there are at least $\sqrt{|\Delta|} \cdot L_{|\Delta|}(\frac{1}{2}, -(1 + c_2(z))/(4z))$ reduced $\mathcal{O}$-ideals with y-smooth norm.*

Proof For any $z > 0$ it is easy to find $\Delta_0 > 0$ and ϵ with $0 < \epsilon \le 1/4$ such that

$$(\log|\Delta|)^{(2+\epsilon)} \le L_{|\Delta|}(\tfrac{1}{2}, z) \le \exp\big((\log(\sqrt{|\Delta|}/2)^{(1-\epsilon)}\big)$$

for any Δ with $|\Delta| > \Delta_0$.

Applying Proposition 4.11 with $x = \sqrt{|\Delta|}/2$, $y = L_{|\Delta|}(\frac{1}{2}, z)$ and the ϵ just found we obtain c_2 such that $\psi_\Delta(x,y) \ge \sqrt{|\Delta|}/2 \cdot L_{|\Delta|}(\frac{1}{2}, -(1+c_2)/(4z))$.

Now note that for each integer a which factors into primes in $\mathcal{P}_y$ and is smaller than $\Delta/2$ there are at least two primitive ideals that factor over $\mathcal{F}_y$ and have norm a. By Lemma 4.10, these ideals are reduced. ∎

In what follows we will let y equal $L_\Delta(\frac{1}{2}, z)$ for some z which we will determine later. Set $\mathcal{F} = \mathcal{F}_y$. We will assume in all that follows that $|\Delta|$ is large enough that Proposition 4.9 assures that $\mathcal{F}$ generates the class group. Ideals that factor over $\mathcal{F}$ will be called smooth. The cardinality of $\mathcal{F}$ will be denoted by n.

In order to derive a lower bound for the probability that a randomly chosen reduced ideal is smooth we need bounds for the total number of reduced ideals in the given order. We will first establish bounds for the number of reduced ideals in each class, and then for the class number.

Lemma 4.13 *If $\Delta < 0$, then there are at most two reduced ideals in each class. If the class is not ambiguous, then there is only one. If there are two, then they are conjugated.*

PROOF See, e.g., [BW88] where the proof is omitted, but can be easily obtained from the preceding results. ∎

Lemma 4.14 *If $\Delta > 0$, then there are at least $2R/\log\Delta$, and at most $2R/\log 2$ reduced ideals in each class.*

PROOF This follows from Proposition 2.13, Lemma 2.7, and Lemma 2.8. ∎

In order to bound the class number we use the analytic class number formula, and bounds for the size of the special value of the Dirichlet L-function. If $\Delta < -4$, then the class number formula reads

$$h_\Delta = \frac{\sqrt{|\Delta|}}{\pi} L(1,\chi) \tag{4.7}$$

Schur proved in [Sch18]

$$L(1,\chi) < \tfrac{1}{2}\log|\Delta| + \log\log|\Delta| + 1. \tag{4.8}$$

If $\Delta > 0$, then the class number formula reads

$$h_\Delta R_\Delta = \sqrt{\Delta} L(1,\chi). \tag{4.9}$$

Schur's result for $\Delta > 0$ was improved by Hua [Hua42] to

$$L(1,\chi) < \tfrac{1}{2}\log\Delta + 1. \tag{4.10}$$

In conclusion, we see that a fraction of $1/L(\frac{1}{2}, 1/(4z)+o(1))$ of all reduced ideals is smooth.

How can we ensure that the ideals occurring in step 3 of the index calculus method described above are chosen in a nearly uniform manner?

If $\Delta < 0$, then Lemma 4.13 shows that smoothness of reduced representatives is a property of the class. (Note that for two conjugate ideals one is smooth, if and only if the other one is.) Thus it suffices to show that drawing $\mathbf{v}_1$ uniformly from some set B will yield nearly uniform distribution of $\bar{\phi}_{\mathcal{F}}(\mathbf{v}_1)$.

If $\Delta > 0$ there are several reduced ideals in each class. Their number varies depending on the class, but only by a factor linear in $\log\Delta$. Again, drawing $\mathbf{v}_1$ uniformly from some set B will yield nearly uniform distribution of $\bar{\phi}_{\mathcal{F}}(\mathbf{v}_1)$. Beyond that, we will have to show how to draw a reduced ideal in the class obtained in a nearly uniformly manner. This will be done in Section 4.6.

4.5 Random ideal classes

In this section we will define a set $B \subset \mathbb{Z}^n$ such that random choice of $\mathbf{v} \in B$ leads to nearly uniform distribution of $\bar{\phi}_{\mathcal{F}}(\mathbf{v})$ in Cl_Δ. For any $l, m \in \mathbb{N}$ we define $B_l(m)) \subset \mathbb{Z}^l$ to be the cube $\{\, (v_i) \in \mathbb{Z}^l \mid \forall i \,:\, 0 \le v_i \le m \,\}$.

Let $\mathcal{G}$ be a generating set for the class group of $\mathcal{O}$. We begin by showing how to choose random ideal classes in Cl_Δ in a nearly uniform manner by computing random power products over $\mathcal{G}$.

Lemma 4.15 *For any lattice $\mathcal{L} \subset \mathbb{Z}^l$ with $\det \mathcal{L} = h$, any $x \in \mathbb{Z}^l$, and any $m \in \mathbb{N}$ we have*

$$\frac{1}{h} \cdot (1 - \frac{h-1}{m}) \le \frac{\mathrm{card}((x + B_l(m)) \cap \mathcal{L})}{\mathrm{card}\, B_l(m)} \le \frac{1}{h} \cdot (1 + \frac{h-1}{m})$$

Proof The lower bound is lemma 4.5 in [Sey87]. The upper bound follows from analogous arguments. See also Lemma 5.1 from [LP92]. ∎

Thus we could generate random ideal classes with the following procedure given a generating set $\mathcal{G}$ for the class group with cardinality l. As always the resulting ideal class is represented by a reduced ideal.

1. Choose randomly and uniformly $\mathbf{v} \in B_l(|\Delta|)$.
2. **return** $\mathfrak{q} \leftarrow \rho_0(\phi_{\mathcal{G}}(\mathbf{v}))$.

For reasons to be seen in Section 4.8 we slightly modify this procedure by 1. embedding $B_l(\Delta)$ into $\mathbb{Z}^n$ via ι defined in (4.3); and 2. allowing for shifts of $\iota(B_l(\Delta))$ by arbitrary vectors in $\mathbb{Z}^n$.

In order to give a listing for the procedure just described we need to explain how to compute $\rho_0(\phi_{\mathcal{F}}(\mathbf{v}))$ given $\mathbf{v} \in \mathbb{Z}^n$ in polynomial time.

This is straightforward in the imaginary quadratic case. We will call the procedure PowProd. It is shown on the next page. PowProd utilizes possible sparseness of its argument.

Lemma 4.16 ($\Delta < 0$). *For arguments $\mathbf{v} = (v_1, \ldots, v_n) \in \iota(\mathbb{Z}^n)$, the algorithm* PowProd *computes a reduced representative of $\bar{\phi}_{\mathcal{F}}(\mathbf{v})$ in time $O(k \cdot \log \max(\, |v_i| \mid i = 1, \ldots, k\,) \cdot (\log|\Delta|)^2)$ where $k = \mathrm{card}\{\, i \mid v_i \neq 0 \,\}$.*

Proof This lemma follows directly from the analysis of the binary exponentiation algorithm, and the time bound for the composition of two reduced ideals given in Corollary 2.5. ∎

As we have seen in 4.3 in the real quadratic situation we would like to obtain for any $\mathbf{v} \in \mathbb{Z}^n$ not only the ideal class $\bar{\phi}_{\mathcal{F}}(\mathbf{v})$ given by a reduced representative $\mathfrak{q}$, but also a generator α of $\mathfrak{q}$ relative to $\phi_{\mathcal{F}}(\mathbf{v})$.

Algorithm 4.1: Power product in the class group

Input: Generating set $\mathcal{F} = (\mathfrak{a}_1, \ldots, \mathfrak{a}_n)$ for the class group Cl_Δ, exponent vector $v \in \mathbb{Z}^n$

Output: Reduced representative $\mathfrak{q}$ of the class $\bar{\phi}_{\mathcal{F}}(v)$

PowProd$(\mathcal{F}, v)$
1. $\mathfrak{q} \leftarrow 1$.
2. **foreach** i such that $v_i \neq 0$
3. Compute $\mathfrak{a} = \mathfrak{a}_i^{v_i}$ by binary exponentiation, reducing after each multiplication.
4. $\mathfrak{q} \leftarrow \mathfrak{q} \cdot \mathfrak{a}$. Reduce $\mathfrak{q}$.
5. **return** $\mathfrak{q}$.

To this end we modify PowProd in such a way that it records the relative generators occurring at each reduction. Their product is kept in power product representation, see Section 2.2.

If PowProdRQ is called several times with the same generating set and a uniform bound for $\|\mathbf{v}\|_\infty$, then the precomputation steps 1 through 5 obviously need to be executed only once.

Lemma 4.17 $\Delta > 0$. *Let $\mathcal{F}$ be a generating set for Cl_Δ with cardinality n. For arguments $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n$ with positive coordinates, the algorithm* PowProdRQ *computes a reduced representative $\mathfrak{q}$ of $\bar{\phi}_{\mathcal{F}}(\mathbf{v})$ and a generator of $\mathfrak{q}$ relative to $\phi_{\mathcal{F}}(\mathbf{v})$ in power product representation $\big((\gamma, a_\gamma)\big)$ where each γ is given in compact representation with no more than $\max(\log_2(v_i + 1))$ components whose heights do not exceed $\Delta^{3/4}$, and whose denominators do not exceed Δ^2. We have*

$$\sum_\gamma a_\gamma < \sum_i \lceil \log_2(v_i + 1) \rceil \,.$$

It executes in time bounded by $O(k \cdot \log \max(\, v_i \mid i = 1, \ldots, n\,) \cdot \log^2 \Delta)$, where $k = \mathrm{card}\{\, i \mid v_i \neq 0 \,\}$.

Proof We first show correctness. For the $\alpha_{i,j}$ and $\mathfrak{a}_{i,j}$ computed in steps 1 through 4 we have $\mathfrak{a}_{i,j} = \alpha_{i,j} \cdot \mathfrak{a}_{i,j-1}^2$ for $j \geq 1$, and hence

$$\mathfrak{a}_{i,j} = \prod_{l=1}^{j} \alpha_{i,l}^{2^{j-l}} \cdot \mathfrak{a}_i^{2^j}$$

Denote the product by $\beta_{i,j}$. We have $\mathfrak{a}_{i,j} = \beta_{i,j} \cdot \mathfrak{a}_i^{2^j}$, with $\beta_{i,j}$ by definition being given in compact representation $\beta_{i,j} = (1, \alpha_{i,j}, \ldots, \alpha_{i,1})$. During the

Algorithm 4.2: Power product with relative generator

Input: Generating set $\mathcal{F} = (\mathfrak{a}_1, \ldots, \mathfrak{a}_n)$ for the class group Cl_Δ, exponent vector $\mathbf{v} \in \mathbb{Z}^n$

Output: Reduced representative $\mathfrak{q}$ of the class $\bar{\phi}_{\mathcal{F}}(\mathbf{v})$ and $\alpha \in \mathcal{K}$ such that $\mathfrak{q} = \alpha \cdot \phi_{\mathcal{F}}(\mathbf{v})$

PowProdRQ($\mathcal{F}, \mathbf{v}$)
Pre-computation
1. **foreach** i such that $v_i \neq 0$
2. $\quad \mathfrak{a}_{i,0} \leftarrow \mathfrak{a}_i$.
3. $\quad$ **for** $j \leftarrow 1$ **to** $\lceil \log_2(v_i + 1) \rceil - 1$
4. $\quad\quad (\mathfrak{a}_{i,j}, \alpha_{i,j}) \leftarrow \mathrm{Rho0}(\mathfrak{a}_{i,j-1}^2)$.
5. $\quad$ Set $\beta_{i,j} \leftarrow (1, \alpha_{i,j}, \ldots, \alpha_{i,1})$.
Multiplication of terms
6. $\mathfrak{c} \leftarrow (1), \gamma \leftarrow 1$.
7. **foreach** pair (i, j) for which bit j in v_i is one
8. $\quad (\mathfrak{c}, \gamma_{i,j}) = \mathrm{Rho0}(\mathfrak{a}_{i,j}\mathfrak{c})$.
9. $\quad \gamma \leftarrow \gamma_{i,j} \cdot \beta_{ij} \cdot \alpha$.
10. **return** $(\mathfrak{c}, \gamma)$.

rest of the algorithm the $\mathfrak{a}_{i,j}$ are multiplied together according to the binary expansion of the exponents v_i, reducing after each multiplication. For $\mathfrak{c}$ and γ returned by PowProdRQ we have

$$\mathfrak{c} = \prod_{i,j:v_{i,j}\neq 0} \gamma_{i,j} \cdot \prod_{i,j:v_{i,j}\neq 0} \beta_{i,j} \cdot \mathfrak{a}_i^{2^j} = \gamma \cdot \phi_{\mathcal{F}}(\mathbf{v}).$$

Each factor $\gamma_{ij}\beta_{ij}$ of γ has no more than $\lceil \log_2(v_i + 1) \rceil$ components by construction. Lemma 2.18 shows that the height of each component is bounded by $\Delta^{3/4}$, and its denominator by Δ^2 since the norm of each ideal reduced in the course of the execution of PowProdRQ is smaller than Δ. The sum of the exponents in the power product representation of γ equals the number of non-zero bits in $\mathbf{v}$, and is hence bounded by $\sum_i \lceil \log_2(v_i + 1) \rceil$, as claimed.

The run-time of the algorithm is dominated by the time spent computing products of two reduced ideals followed by reduction. At most $2k \cdot \sum_i \lceil \log_2(v_i + 1) \rceil$ such computations are performed, each requiring time $O(\log^2 \Delta)$ due to Proposition 2.4. This yields the run-time bound given in the lemma. ∎

The procedure for selecting random ideal classes in an order is now obvious. We just give the variant for $\Delta < 0$.

Algorithm 4.3: Random ideal class from power product (IQ)

Input: Ideal sets $\iota : \mathcal{G} \hookrightarrow \mathcal{F} \subset \mathcal{R}_\Delta$ generating Cl_Δ;
displacement vector $\mathbf{w} \in \mathbb{Z}^n$ where $n = \operatorname{card} \mathcal{F}$
Output: random ideal class represented by $\mathfrak{q} \in \mathcal{R}_\Delta$

RandClass$(\mathcal{G}, \mathcal{F}, \mathbf{w})$
1. Choose randomly and uniformly $\mathbf{v} \in B_l(\Delta)$ where $l = \operatorname{card} \mathcal{G}$.
2. **return** $\mathfrak{q} \leftarrow$ PowProd$(\mathcal{F}, \iota(\mathbf{v}) + \mathbf{w})$.

Lemma 4.18 *For any given ideal class $\bar{\mathfrak{a}} \in \mathrm{Cl}_\Delta$, the probability that algorithm* RandClass *returns an ideal representing $\bar{\mathfrak{a}}$ exceeds $(|\Delta| - h)/(h \cdot |\Delta|)$.*

Proof This follows immediately from Lemma 4.15. ∎

Let w be the random variable taking uniformly distributed values in $B_l(\Delta) \subset \mathbb{Z}^l$. Then the behavior of RandClass on input of sets $\iota : \mathcal{G} \hookrightarrow \mathcal{F}$, and displacement vector $\mathbf{w}_0$ is modeled by the push-forward $\mathfrak{w}$ of w with values in the set $\mathcal{R}_\Delta$ of reduced ideals defined by $\mathfrak{w} = \rho_0(\phi_\mathcal{F}(\iota(w) + \mathbf{w}_0))$. By passing from ideals to classes, we also define $\bar{\mathfrak{w}}$ with values in Cl_Δ.

The real quadratic variant RandClassRQ is obtained from RandClass by using PowProdRQ to obtain the relative generator of the reduced ideal $\mathfrak{q}$ returned relative to $\phi_\mathcal{F}(\mathbf{v})$.

Let w be the random variable taking uniformly distributed values in $B_k(\Delta) \subset \mathbb{Z}^k$. Then RandClassRQ is modeled by the push-forward $\tilde{\mathfrak{w}} = (\mathfrak{w}, \omega)$ of w with values in $\mathcal{R} \times \mathcal{K}$ defined by $\mathfrak{w} = \rho_0(\phi_\mathcal{G}(w)) = \omega \cdot \phi_\mathcal{G}(w)$ and $|\operatorname{Log} \omega| < \frac{1}{2} \log \Delta$.

4.6 Random ideals on a given cycle

If $\mathcal{O}$ is imaginary quadratic, then choosing a random ideal class is essentially equivalent to choosing a random reduced ideal, since there is either only one such ideal or two conjugated ones in each class. If $\mathcal{O}$ is real quadratic, however, there are many reduced ideals in each class. In order to find with sufficient probability a smooth reduced ideal we need not only randomize among ideal classes but also among the reduced ideals in a given class. This section explains how this is done.

Let $\mathfrak{a} \in \mathcal{R}_\Delta$ be some reduced $\mathcal{O}$-ideal. For any $d \in \mathbb{N}$ we define the set

$$S_d = S_d(\mathfrak{a}) = \{\, (\mathfrak{b}, \alpha) \mid \mathfrak{b} \text{ is reduced}, \mathfrak{b} = \alpha\mathfrak{a}, d \le \operatorname{Log} \alpha / \log \Delta < (d+1) \,\}.$$

Reduced ideals are approximately evenly distributed among the sets S_d if d is allowed to vary over a large interval.

Lemma 4.19 *Let* $\mathfrak{a}$ *and* $\mathfrak{b}$ *be two equivalent reduced ideals. Then for any* $M \in \mathbb{N}$*, we have* $\operatorname{card}\{\, d \mid 0 \le d \le M \text{ and } \exists\, \alpha \in \mathcal{K} : (\mathfrak{b}, \alpha) \in S_d(\mathfrak{a}) \,\} = M \log \Delta / R_\Delta + e$ *with* $0 \le e < 2$*.*

Proof Let $\mathfrak{b} = \alpha_0 \mathfrak{a}$, where α_0 is chosen such that $0 \le \operatorname{Log} \alpha_0 < R$. Then for any $k \in \mathbb{Z}$ the pair $(\mathfrak{b}, \alpha_0 \varepsilon_\Delta^k)$ is in $S_d(\mathfrak{a})$ if and only if $d \log \Delta \le \operatorname{Log} \alpha + k R_\Delta < (d+1) \log \Delta$. This is equivalent to

$$d = \left\lfloor \frac{\operatorname{Log} \alpha_0 + k R_\Delta}{\log \Delta} \right\rfloor .$$

Call the expression on the right $d(k)$. If k lies in the interval $[0, M \log \Delta / R_\Delta - 1]$, then $d(k)$ lies in the interval $[0, M]$. On the other hand, if $d(k) \in [0, M]$, then $0 \le k < M \log \Delta / R_\Delta + 1$. ∎

Fix some $M \gg R$. The preceding lemma leads us to proceed as follows if we want to produce a random reduced ideal in the cycle with representative $\mathfrak{a}$.

1. Choose some random $d \in [0, M]$.
2. Enumerate all elements in $S_d(\mathfrak{a})$.
3. Choose randomly among them.

In order to estimate the probability that a given reduced ideal is found by the procedure we need to know the cardinality of each S_d. This is a trivial consequence of lemmata 2.7 and 2.8.

Lemma 4.20 *Fix* $\mathfrak{a} \in \mathcal{R}_\Delta$*. Let* $d \ge 0$*. Then* $2 \le \operatorname{card} S_d(\mathfrak{a}) \le 2 \log_2 \Delta$*.*

Next, we show how to enumerate all elements of $S_d(\mathfrak{a})$ given reduced $\mathfrak{a} \in \mathcal{R}_\Delta$ and $d \in \mathbb{N}$ in time polynomial in $\log d$ and $\log \Delta$. For this to be possible, the field elements belonging to each pair contained in S_d need to be given in compact representation.

We begin by showing how to find for any d some α (in compact representation) such that (α) is reduced, and $\operatorname{Log} \alpha \approx d \cdot \log \Delta$. This is achieved by algorithm Prox, see below, which is completely analogous to Algorithm 4.2 in [BTW95].

Lemma 4.21 *Given discriminant* Δ*, distance* $x \in \mathbb{Q}$*, and error bound* ϵ *such that* $0 < \epsilon < 1$*,* Prox *computes in time polynomial in* $\log \Delta$*,* $\log x$ *and* $\log \epsilon$ *a triple* $(\mathfrak{a}, \alpha, \delta))$ *consisting of a reduced ideal* $\mathfrak{a}$*, integral* $\alpha \in \mathcal{O}$ *in compact representation* $(1, \gamma_1, \ldots, \gamma_k)$ *and* $\delta \in \mathbb{Q}$ *with the following properties:*

The ideal $\mathfrak{a}$ *is generated by* α*. The number* k *of components of* α *is bounded by* $O(\log x)$*. Each component* γ_j *has height bounded by* $\sqrt{3}\Delta^{5/4}$ *and denominator bounded by* Δ*. The number* δ*, finally, is a small rest distance satisfying* $0 \le |\operatorname{Log} \alpha + \delta - x| < \epsilon$ *and* $0 \le \delta < \left\lceil \frac{1}{2} \log \Delta \right\rceil$*.*

Algorithm 4.4: Reduced ideal with generator of given length

Input: Discriminant Δ, $x \in \mathbb{Q}$, error bound ϵ

Output: $\mathfrak{a} \in \mathcal{R}_\Delta$ in standard representation,
α in compact representation such that $\mathfrak{a} = (\alpha)$,
$\delta \in \mathbb{Q}$ satisfying
$0 \le \delta < \left\lceil \frac{1}{2} \log \Delta \right\rceil$ and $0 \le |\mathrm{Log}\, \alpha + \delta - x| < \epsilon$

PROX(Δ, x)
1. **if** $x < \left\lceil \frac{1}{2} \log \Delta \right\rceil$
2. **return** $(\mathcal{O}, 1, x)$.
3. **else**
4. *Recurse* $(\mathfrak{a}, \alpha = (1, \gamma_1, \ldots, \gamma_k), \delta) = \mathrm{PROX}(\Delta, x/2, \epsilon/4)$.
5. $\mathfrak{a} \leftarrow \mathfrak{a}^2$, $\alpha \leftarrow \alpha^2$, $\delta \leftarrow 2\delta$.
6. *Reduce* $(\mathfrak{c}, \gamma) \leftarrow \mathrm{CLOSEL}(\mathfrak{a})$.
7. $\mathfrak{a} \leftarrow \mathfrak{c}$, $\gamma_{k+1} = \gamma$, $\alpha \leftarrow \alpha\gamma_{k+1} = (1, \gamma_1, \ldots, \gamma_{k+1})$,
$\delta \leftarrow \delta - \mathrm{LOG}(\gamma, \epsilon/4)$.
Correct δ *by walking* $O(\log \Delta)$ *steps to the right*
8. **while** $\delta \ge \left\lceil \frac{1}{2} \log \Delta \right\rceil$
9. $\gamma_{k+1} \leftarrow \gamma_{k+1}/\gamma(\mathfrak{a})$, $\alpha \leftarrow \alpha/\gamma(\mathfrak{a})$,
$\delta = \delta + \mathrm{LOG}(\gamma(\mathfrak{a}), \epsilon/\left\lceil 12 \log_2 \Delta \right\rceil)$, $\mathfrak{a} \leftarrow \rho(\mathfrak{a})$
10. **return** $(\mathfrak{a}, \alpha = (1, \gamma_1, \ldots \gamma_{k+1}), \delta)$.

PROOF The proof follows that of Theorem 4.1 in [BTW95]. ∎

Thus we may proceed as follows given a reduced ideal $\mathfrak{a}$ and $d \in \mathbb{N}$:

1. Compute $\mathfrak{b} = (\beta)$ and δ with $\operatorname{Log}\beta + \delta \approx d \log \Delta$ and $\delta > 0$.
2. Reduce the product of $\mathfrak{a}$ and $\mathfrak{b}$ to get $\mathfrak{c}_0$ and γ such that $\mathfrak{c}_0 = \gamma \cdot \mathfrak{a}\mathfrak{b}$, and $\operatorname{Log}\gamma_0 < 0$.
3. Walk a few reduction steps to find minimal I such that $\mathfrak{c}_I = \alpha_I \mathfrak{c}_0 = \rho^I(\mathfrak{c}_0)$ and $\operatorname{Log}\alpha_I + \operatorname{Log}\beta + \operatorname{Log}\gamma \geq d \cdot \log \Delta$.
4. Walk a few more reduction steps to find all $\mathfrak{c}_j = \alpha_j \mathfrak{c}_0$ with $\operatorname{Log}\alpha_j + \operatorname{Log}\beta + \operatorname{Log}\gamma_0 < (d+1) \cdot \log \Delta$. Let J be the maximal such index.
5. Return $(\mathfrak{c}_I, \ldots, \mathfrak{c}_J)$.

A detailed listing of the algorithm which we are going to call ENUMS can be found on this page.

Note that we can assure that all reduced ideals in S_d get enumerated, but due to the imprecise computation of logarithms in this enumeration process, the enumeration may inadvertently contain ideals with relative generators from a slightly larger interval.

To be precise, Log always returns values which are larger in absolute value than the correct (irrational) value. Thus the value of I may be one too small. In order to obtain all elements of S_d we have to compare in step 4 not with $(d+1) \cdot \log \Delta$ but some bound larger than $(d+1) \cdot \log \Delta + \epsilon$ where ϵ is the largest error the logarithm computations incur. This may again lead to the inclusion of one ideal too many in the returned list.

Algorithm 4.5: Enumeration of cycle section

Input: Discriminant Δ, reduced ideal $\mathfrak{a} \in \mathcal{R}_\Delta$,
distance parameter $d \in \mathbb{N}$

Output: set of ideals $S = \{\mathfrak{c}_0, \ldots, \mathfrak{c}_i\}$ such that $S_d(\mathfrak{a}) \subset S$ and $\operatorname{card}(S \setminus S_d(\mathfrak{a})) \leq 2$

ENUMS$(\Delta, \mathfrak{a}, d)$
1. Set error bound $\epsilon \leftarrow 1/(2\Delta)$.
2. $(\mathfrak{b}, \beta, \delta) \leftarrow$ PROX$(\Delta, d \log \Delta, \epsilon/4)$.
3. $(\mathfrak{c}_0, \gamma) \leftarrow$ CLOSEL$(\mathfrak{a} \cdot \mathfrak{b})$, and $\delta \leftarrow \delta -$ LOG$(\gamma, \epsilon/4)$.
4. $\alpha_0 \leftarrow 1$ and $I \leftarrow 0$.
5. **for** $(i = 0, \delta > -\log \Delta - \epsilon, i \leftarrow i + 1)$
6. **if** $\delta \geq \epsilon$ **then** $I \leftarrow i$
7. **else** $J \leftarrow i$
8. $\alpha_{i+1} \leftarrow \alpha_i / \gamma(\mathfrak{c}_i)$, $\delta \leftarrow \delta +$ LOG$(\gamma(\mathfrak{c}_i), \epsilon/(9 \log_2 \Delta)$, $\mathfrak{c}_{i+1} \leftarrow \rho(\mathfrak{c}_i)$.
9. **return** $S \leftarrow \{(\mathfrak{c}_I, \alpha_I \beta \gamma), \ldots, (\mathfrak{c}_J, \alpha_J \beta \gamma)\}$

Lemma 4.22 *Given discriminant δ, reduced ideal $\mathfrak{a} \in I_\Delta$, and $d \in \mathbb{N}$,* EnumS *computes in time polynomial in* $\log \Delta$ *and* $\log d$ *a set S such that $S_d(\mathfrak{a}) \subset S$, there are at most two elements in $S \smallsetminus S_d(\mathfrak{a})$ and for any element $(\mathfrak{b}, \beta)$ of S we have $d \log \Delta - \Delta^{-1} < \operatorname{Log} \beta < (d+1) \log \Delta + \Delta^{-1}$. An element of S is a pair $(\mathfrak{c}, \gamma)$ consisting of a reduced ideal $\mathfrak{c}$, and a quadratic number γ given in compact representation with no more than $\log d \log\log \Delta$ components. Each component has height not exceeding $3\Delta^{9/2}$, and denominator not exceeding Δ.*

Proof We begin by giving a bound for the number of iterations executed in the loop (steps 5 through 8). Due to Lemma 2.8 we know that δ is reduced during two consecutive iterations by at least $\log 2 - 2\tilde{\epsilon}$ where $\tilde{\epsilon} = \epsilon/(9 \lceil \log_2 \Delta \rceil)$ is the the maximal error incurred in the Log computation. After step 3, the distance variable δ is bounded by $\left\lceil \frac{1}{2} \log \Delta \right\rceil + \frac{1}{2} \log \Delta$. This follows from Lemma 4.21, Lemma 2.19, and Lemma 2.23. It is now easy to verify that after no more than $4(\log_2 \Delta + 2)$ iterations δ will become smaller than $-\log \Delta - \epsilon$.

From this it follows that the maximal error, i.e., the maximum difference between $\operatorname{Log} \alpha_i \beta \gamma$ and $d \log \Delta - \delta$ does not exceed ϵ in the course of execution of EnumS. Hence, we obtain $\operatorname{Log} \alpha_I \beta \gamma > d \log \Delta - 2\epsilon$, and $\operatorname{Log} \alpha_J \beta \gamma < (d+1) \log \Delta + 2\epsilon$. This implies $S \smallsetminus S_d(\mathfrak{a}) \subset \{\mathfrak{c}_I, \mathfrak{c}_J\}$.

Finally, we conclude from $\operatorname{Log} \beta\gamma < d \log \Delta + \epsilon/4$, and $\operatorname{Log} \alpha_{J+1} \beta\gamma > (d+1) \log \Delta$ using Proposition 2.13, and Lemma 2.7 that $S_d(\mathfrak{a}) \subseteq S$.

It is clear from Lemma 4.21, Proposition 2.4, Lemma 2.2 and the bound on the number of iterations obtained above that EnumS executes in polynomial time.

The bound on the number of components of each relative generator $\alpha_i \beta \gamma$ returned follows from Lemma 4.21. This lemma also implies the bounds for height and denominator for all but the last component of this product. This last component is $\alpha_i \beta$. Applying the same reasoning as we did in the proof of Lemma 2.18 and using the fact that $0 < \operatorname{Log} \alpha_i \beta < 2 \log \Delta + 2$, we obtain the height and denominator bounds given in the theorem. ∎

Thus we can write down the sought procedure for the choice of a random reduced ideal in an ideal class given by some reduced representative. This procedure will be called RandRed and is shown on the next page.

We summarize the properties of RandRed in the following proposition.

Proposition 4.23 *Let $\mathfrak{a}$ be a given reduced $\mathcal{O}$-ideal.* RandRed *computes randomly in polynomial time some reduced $\mathcal{O}$-ideal $\mathfrak{b}$, and $\beta \in \mathcal{K}$ in compact representation such that $\mathfrak{b} = \beta \cdot \mathfrak{a}$ is reduced.*

For any reduced $\mathfrak{b}$ equivalent to $\mathfrak{a}$ the probability that RandRed *outputs $\mathfrak{b}$ on input $\mathfrak{a}$ is contained in the interval* $(\log(2)/(2R) - 1/\Delta, \log \Delta/(2R) + 1/\Delta)$.

Algorithm 4.6: Random choice of reduced ideal in given class

Input: Discriminant Δ, reduced ideal $\mathfrak{a} \in I_\Delta$

Output: Reduced ideal $\mathfrak{b}$ and relative generator β in compact representation such that $\mathfrak{b} = \beta \cdot \mathfrak{a}$

RandRed$(\Delta, \mathfrak{a})$

1. Choose randomly and uniformly some d in $[0, \Delta]$.
2. Get $S \leftarrow$ EnumS$(\Delta, \mathfrak{a}, d)$.
3. Choose randomly and uniformly a pair $(\mathfrak{b}, \beta)$ from S.

Moreover, the probability that the second component β of the output of RandRed *fulfills* $\operatorname{Log}\beta \in [eR, (e+1)R)$ *conditional on the fact that the first component is some fixed $\mathfrak{b}$ is equal to $1/N$ where N is bounded from below by $\Delta \cdot \log(2)/R - 2/\log_2 \Delta$ and from above by $\Delta \cdot \log(2) \log^2 \Delta/R + \log_2 \Delta$ if $e < (\Delta \log \Delta - R)/R$.*

Proof Run-time bound for and correctness of the output format of RandRed follow immediately from Lemma 4.22.

We will model the behavior of RandRed with a random variable $\tilde{\mathfrak{r}} = (\mathfrak{r}, \xi)$, where the component $\mathfrak{r}$ takes values in the set $\mathcal{R}_\Delta$ of reduced variables, and ξ takes values in $\mathcal{K}$. It is parameterized by $\mathfrak{a}$ ranging over $\mathcal{R}_\Delta$.

Let d be the random variable taking uniformly distributed values in $[0, \Delta]$. Let further i be the random variable taking uniformly distributed values in $[1, \lfloor 2\log_2 \Delta \rfloor !]$. Introduce the natural ordering in $S_d(\mathfrak{a})$. Let $s_d = \operatorname{card} S_d(\mathfrak{a})$ and $(\mathfrak{b}_{d,i}, \beta_{d,i})$ be the i-th element of $S_d(\mathfrak{a})$. Then we obtain $\tilde{\mathfrak{r}}(\mathfrak{a})$ from d and i by setting $\mathfrak{r} = \mathfrak{b}_{d,i \bmod s_d}(\mathfrak{a})$ and $\xi = \beta_{d,i \bmod s_d}$. Thus $\mathfrak{r}$ and ξ taken individually are correlated by the relation $\mathfrak{r}(\mathfrak{a}) = \xi(\mathfrak{a}) \cdot \mathfrak{a}$.

Thus, we have for any pair $(\mathfrak{b}, \beta)$ with $\mathfrak{b} = \beta \cdot \mathfrak{a}$ where $\mathfrak{b}$ is reduced and $d \le \operatorname{Log}\beta / \log \Delta < d + 1$

$$\operatorname{Prob}(\tilde{\mathfrak{r}} = (\mathfrak{b}, \beta)) = \frac{1}{\Delta \cdot s_d} \tag{4.11}$$

with $s_d = \operatorname{card} S_d(\mathfrak{a})$. Hence, it follows from Lemma 4.20

$$\frac{1}{\lfloor 2\log \Delta \rfloor \cdot \Delta} \le \operatorname{Prob}(\tilde{\mathfrak{r}} = (\mathfrak{b}, \beta)) \le \frac{1}{2\Delta}. \tag{4.12}$$

The first component $\mathfrak{r} = \mathfrak{r}(\mathfrak{a})$ takes values in the cycle $\mathcal{C}(\mathfrak{a})$ of reduced ideals in the class of $\mathfrak{a}$. From (4.12) it is easy to deduce bounds for $\operatorname{Prob}(\mathfrak{r} = \mathfrak{b})$ for fixed $\mathfrak{b}$. Indeed, we obtain

$$\operatorname{Prob}(\mathfrak{r} = \mathfrak{b}) = \frac{1}{\Delta} \cdot \sum_{d\,:\,\mathfrak{b} \in S_d(\mathfrak{a})} \frac{1}{s_d}. \tag{4.13}$$

where the sum is over all d in $[0, \Delta]$ with $\mathfrak{b} \in S_d(\mathfrak{a})$. Using Lemma 4.19 and Lemma 4.20 this yields

$$\log(2)/(2R) - 1/\Delta < \mathrm{Prob}(\mathfrak{r} = \mathfrak{b}) < \log \Delta/(2R) + 1/\Delta \tag{4.14}$$

This proves the second claim of the proposition.

From $\mathfrak{r}$ and ξ we derive a third random variable e_{cyc} that captures the probability that the second component of the output of RandRed has length in $[eR, (e+1)R)$ conditional on the first component being some fixed $\mathfrak{b}$. It is parameterized by pairs of equivalent reduced ideals $\mathfrak{a}, \mathfrak{b}$. Fix such a pair. e_{cyc} has range in $[0, (\Delta \log \Delta + 1)/R)$ and is defined by

$$\mathrm{Prob}(e_{\mathrm{cyc}}(\mathfrak{a}, \mathfrak{b}) = e) = \frac{\mathrm{Prob}(\mathfrak{r}(\mathfrak{a}) = \mathfrak{b}, eR \leq \mathrm{Log}\, \xi(\mathfrak{a}) < (e+1)R)}{\mathrm{Prob}(\mathfrak{r}(\mathfrak{a}) = \mathfrak{b})} \tag{4.15}$$

From (4.11) and (4.13) we deduce

$$\mathrm{Prob}(e_{\mathrm{cyc}} = e) = 1/N, \text{ where } N = \sum_{d\,:\,\mathfrak{b} \in S_d(\mathfrak{a})} \frac{s}{s_d}, \tag{4.16}$$

and $s = s_d$ with d chosen such that the unique β for which $\mathfrak{b} = \beta\mathfrak{a}$ and $eR \leq \mathrm{Log}\, \beta < (e+1)R$ satisfies $d \leq \mathrm{Log}\, \beta / \log \Delta < (d+1)$. If $e < ((\Delta + 1) \log \Delta - R)/R$ then this d is smaller or equal to Δ. Applying Lemma 4.19 (for the number of the summands) and Lemma 4.20 (for the size of the summands) we obtain the bound for N given in the last claim of the proposition

$$\Delta \cdot \log(2)/R - 2/\log_2 \Delta < N < \Delta \cdot \log(2) \log^2(\Delta)/R + \log_2 \Delta \tag{4.17}$$

■

Note 4.24 Property (4.14) depends only on the class of $\mathfrak{a}$, not on $\mathfrak{a}$ itself. Thus we will sometimes suppress the dependence of $\mathfrak{r}$ on the representative of the class.

Likewise, property (4.17) does not depend on the pair $(\mathfrak{a}, \mathfrak{b})$ parameterizing e_{cyc}. Thus we will also suppress the dependency of e_{cyc} on $\mathfrak{a}$ and $\mathfrak{b}$ wherever we will only need the range of e_{cyc} and said property.

Note 4.25 From Lemma 4.22 we see that RandRed differs slightly from the model. The probability that a particular pair is output by RandRed is bounded from below by $\Delta^{-1} \cdot (s_d + 2)^{-1}$ and from above by $\Delta^{-1} \cdot (s_d^{-1} + s_{d+1}^{-1})$ with suitably chosen d. This differs from the value given in (4.11) by a factor of at most $\frac{1}{2}$ from below and $1 + \log_2 \Delta$ from above. The analysis that builds on Proposition 4.23 will not be significantly impacted by terms polynomial in $\log \Delta$. Thus we will ignore the divergence of RandRed from the model used in the proof. It will be easy for the reader to re-introduce the corrective factors if need be.

4.7 Potential relations

We call a pair $(\mathbf{v}, \mathfrak{a}) \in \mathbb{Z}^n \times \mathcal{R}_\Delta$ a *potential relation* if $\mathfrak{a} \in \bar{\phi}_\mathcal{F}(\mathbf{v})$. In the real quadratic case, an *extended potential relation* is a triple $(\mathbf{v}, \mathfrak{a}, \alpha) \in \mathbb{Z}^n \times \mathcal{R}_\Delta \times \mathcal{K}$ if $\mathfrak{a} = \alpha \cdot \phi_\mathcal{F}(\mathbf{v})$. From 4.3 it is clear that each potential relation yields a relation in $\mathcal{L}_\mathcal{F}$, or $\tilde{\mathcal{L}}_\mathcal{F}$, respectively, if and only if $\mathfrak{a}$ factors over $\mathcal{F}$.

The preceding two sections yield an algorithm for the generation of (extended) potential relations, listed on the following page, which will be called PotRelIQ and PotRelRQ depending on the sign of Δ. The use of an offset $\mathbf{w}$ in the input will be explained in Section 4.8. We will use the shorthand PotRel when we refer to both of these algorithms.

Lemma 4.26 PotRel *executes in polynomial time.* PotRelIQ *returns potential relations.* PotRelRQ *returns extended potential relations.*

Proof We obtain the run-time of PotRel from Lemma 4.16, Lemma 4.17, and Proposition 4.23.

PotRel returns potential relations due to Lemma 4.16.

By Lemma 4.17 we have $\mathfrak{a} = \alpha \cdot \phi_\mathcal{F}(\mathbf{v})$ after step 2 of PotRelRQ. By Proposition 4.23 we have $\mathfrak{b} = \beta \cdot \mathfrak{a}$ after step 3. It follows that the returned triple $(\mathbf{v}, \mathfrak{a}, \alpha\beta)$ is an extended potential relation. ∎

We define random variables which model PotRelIQ and PotRelRQ, respectively.

Recall that w was defined to be the random variable taking uniformly distributed values in $B_k(\Delta)$. The following variables were defined on the basis of w: the random variable $\mathfrak{w} = \phi_\mathcal{F} \circ \iota(w)$ taking values in $I_\mathcal{F}$, and $\mathfrak{w}_0 = \rho_0(\mathfrak{w})$ taking values in the set $\mathcal{R}_\Delta$ of reduced ideals in $\mathcal{O}$. Further we defined the map ω_0 from $I_\mathcal{F}$ to $\mathcal{K}$ with the properties $\rho_0(\mathfrak{a}) = \omega_0(\mathfrak{a}) \cdot \mathfrak{a}$ and $|\operatorname{Log} \omega_0(\mathfrak{a})| < \frac{1}{2} \log \Delta$, and the corresponding random variable $\omega = \omega_0(\mathfrak{w})$.

Then PotRelIQ is modeled by $(w, \mathfrak{w}_0)$.

Recall further the random variable $\tilde{\mathfrak{r}} = (\mathfrak{r}, \xi)$ modeling RandRed defined in the previous section. PotRelRQ is modeled by $(w, \mathfrak{r}(\mathfrak{w}_0), \xi(\mathfrak{w}_0) \cdot \omega(\mathfrak{w}))$.

Proposition 4.27 *For any $z > 0$ there exist $\Delta_0(z) > 0$ and a function $c_3(z, \Delta) > 0$ tending to 0 with increasing $|\Delta|$ that satisfy the following property.*

Let Δ be a fundamental discriminant with $|\Delta| > \Delta_0(z)$. Let further $\mathcal{G}$ be a generating set for Cl_Δ with $\operatorname{card} \mathcal{G} \leq (\log \Delta)^{9/4}$, and $\mathcal{G} \subset \mathcal{F}_y$ where $y = L_\Delta(\frac{1}{2}, z)$. Then the probability that a call to PotRel *with input Δ, $\mathcal{G}$, and $\mathcal{F} = \mathcal{F}_y$ yields an ideal $\mathfrak{c}$ that factors over $\mathcal{F}$ is bounded from below by $L_{|\Delta|}(\frac{1}{2}, -(1 + c_3(z, \Delta))/(4z))$*

Proof The sought probability is bounded from below by a product of two factors: 1. a lower bound for the probability that a particular reduced ideal

Algorithm 4.7: Generation of potential relation (IQ)

Input: Discriminant Δ, factor base $\mathcal{F}$ with $\operatorname{card}\mathcal{F} = n$, generating set $\mathcal{G} \subset \mathcal{F}$ with $\operatorname{card}\mathcal{G} = k$ offset $\mathbf{w} \in \mathbb{Z}^n$
Output: Potential relation $(\mathbf{v}, \mathfrak{c})$

PotRelIQ$(\Delta, \mathcal{F}, \mathcal{G}, \mathbf{w})$
1. Choose randomly and uniformly $\mathbf{v} \in \mathbf{w} + \iota(B_k(\Delta))$.
2. Let $\mathfrak{c} \leftarrow$ PowProd$(\mathcal{F}, \mathbf{v})$.
3. **return** $\mathbf{v}, \mathfrak{c}$.

Algorithm 4.8: Generation of potential relation (RQ)

Input: Discriminant Δ, factor base $\mathcal{F}$ with $\operatorname{card}\mathcal{F} = n$, generating set $\mathcal{G} \subset \mathcal{F}$ with $\operatorname{card}\mathcal{G} = k$, offset $\mathbf{w} \in \mathbb{Z}^n$
Output: Extended potential relation $(\mathbf{v}, \mathfrak{c}, \gamma)$

PotRelRQ$(\Delta, \mathcal{F}, \mathcal{G}, \mathbf{w})$
1. Choose randomly and uniformly $\mathbf{v} \in \mathbf{w} + \iota(B_k(\Delta))$.
2. Let $(\mathfrak{a}, \alpha) \leftarrow$ PowProdRQ$(\mathcal{F}, \mathbf{v})$.
3. Let $(\mathfrak{b}, \beta) \leftarrow$ RandRed$(\Delta, \mathfrak{a})$.
4. Let $\mathfrak{c} \leftarrow \mathfrak{b}$ and $\gamma \leftarrow \alpha\beta$.
5. **return** $\mathbf{v}, \mathfrak{c}, \gamma$.

is returned by PotRel; and 2. the number of y-smooth reduced ideals. The first factor is given by (IQ) Lemma 4.18, or (RQ) this lemma in combination with Proposition 4.23. Its main term is $1/h$ or $\log(2)/(2hR)$, respectively. Corollary 4.12 gives the second factor. Applying the class number formula (4.7), or (4.9), respectively, and the bounds for the special value of the L-function (4.8), or (4.10) yields the desired result. ∎

Note: We will henceforth assume that PotRel is always called with input satisfying the assumptions of Proposition 4.27.

4.8 Generating full rank relation lattices

In this section we will show how to obtain a set of relations generating a lattice $\mathcal{M} \subset \mathcal{L}_{\mathcal{F}}$ which has full rank. The idea for the procedure appeared already in Seysen's work [Sey87].

We will generate an ordered set $\mathcal{V}$ of relations such that $A(\mathcal{V})$ is diagonally dominant. This implies that $\mathcal{M} = \langle \mathcal{V} \rangle$ has full rank. In order to obtain the i-th member of this set we call PotRel with offset $\mathbf{w} = 2n\Delta \cdot \mathbf{e}_i$.

The listing of algorithm FullRkRL for $\Delta < 0$ can be found on the current page. The algorithm for $\Delta > 0$ is obtained from that algorithm by saving the relative generator returned by PotRelRQ, and printing it together with the other data in step 16 for every smooth potential relation found.

Algorithm 4.9: Generation of a full rank relation lattice (IQ)

Input: Discriminant Δ, factor base $\mathcal{F} = \mathcal{F}_y$ with $\operatorname{card}\mathcal{F} = n$, generating set $\mathcal{G} \subset \mathcal{F}$

Output: relation vectors $\mathbf{u}_i$, $i = 1, \ldots, n$ with $\mathbf{u}_{i,i} > 2n|\Delta| - \log_2|\Delta|$ and $-\log_2|\Delta| < u_{i,j} \le |\Delta|$

FullRkRL$(\Delta, \mathcal{F}, \mathcal{G})$

1. Set $S \leftarrow (1 + \log_2 n)/2 \cdot \max(2, L_{|\Delta|}(\frac{1}{2}, (1 + c_3(z, \Delta))/(4z)))$.
2. **for** $i \leftarrow 1$ **to** n
3. **for** $j \leftarrow 1$ **to** $\lceil S \rceil$
4. $(\mathbf{v}_j, \mathfrak{a}_j) \leftarrow$ PotRelIQ$(\Delta, \mathcal{F}, \mathcal{G}, 2n\Delta \cdot \mathbf{e}_i)$.
5. $\mathcal{N}_j \leftarrow \{\, p \in \mathcal{P}_y \mid p \mid \mathrm{N}(\mathfrak{a}_j) \,\}$ for $j = 1, \ldots, S$ (computed using Algorithm 7.1 from [Ber02]).
6. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{Z}^n$.
7. **for** $j \leftarrow 1$ **to** $\lceil S \rceil$
8. Initialize $q \leftarrow 1$.
9. **foreach** $p \in \mathcal{N}_j$
10. Find e_p such that $p^{e_p} \parallel \mathrm{N}(\mathfrak{a}_j)$.
11. $q \leftarrow q \cdot p^{e_p}$.
12. **if** $q = \mathrm{N}(\mathfrak{a}_j)$ i.e. $\mathfrak{a}_j$ is y-smooth
13. **foreach** $p \in \mathcal{N}_j$
14. Find $\mathfrak{p}$ with $N(\mathfrak{p}) = p$ such that $\mathfrak{p}\mathfrak{a}_j$ is primitive.
15. Set $\mathbf{w} \leftarrow \mathbf{w} + e_p \cdot \phi_{\mathcal{F}}^{-1}(\mathfrak{p})$.
16. **print** $\mathbf{u}_i \leftarrow \mathbf{v}_j - \mathbf{w}$.
17. **break**
18. **if** no $\mathfrak{a}_j$ is y-smooth **then print** Failure and **exit**

Proposition 4.28 *Fix $z > 0$, and some $c > 0$. Assume we are given a factor base $\mathcal{F} = \mathcal{F}_y$ of cardinality $n \ge 7$ where $y = L_{|\Delta|}(\frac{1}{2}, z)$. Let L equal $L_{|\Delta|}(\frac{1}{2}, (1 + c_3(z, \Delta))/(4z))$. Assume further that $\mathcal{G}$ is a subset of $\mathcal{F}$ such that $\bar{\phi}_{\mathcal{F}}(\mathcal{G}) = \mathrm{Cl}_\Delta$ and $\operatorname{card}\mathcal{G} \le c \log^2|\Delta|$.*

Then, Algorithm FullRkRL *succeeds with probability exceeding $\frac{1}{3}$ and prints a sequence $\mathcal{V} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ of relations in $\mathcal{L}_{\mathcal{F}}$ for which $A(\mathcal{V})$ is*

diagonally dominant. Otherwise it announces failure.

The running time required by FullRkRL *is bounded by* $O^\sim(n \cdot L)$*, and space by* $L_{|\Delta|}(\frac{1}{2}, (1+c_4(z,\Delta)/(4z))$ *where for any* $z > 0$ *the function* $c_4(z,\Delta)$ *tends to* 0 *with growing* $|\Delta|$*.*

Proof We begin by analyzing the probability that FullRkRL succeeds. Let L denote $\max(2, L_{|\Delta|}(\frac{1}{2}, (1 + c_3(z,\Delta))/(4z)))$ where c_3 is the function from Proposition 4.27. Let further $l = 1 + \log_2 n$ and S denote—as it does in the algorithm—the quantity $l/2 \cdot L$.

Then it follows from Proposition 4.27 that the probability that in $\lceil S \rceil$ calls to PotRel no smooth potential relation is returned is smaller than $(1 - 1/L)^S$. Now

$$(1 - 1/L)^{lL/2} \leq 2^{-l} < 1/n,$$

since $L \geq 2$. Hence the probability that for all i the main loop produces a relation is greater than $(1 - 1/n)^n$ which exceeds $1/3$ for $n \geq 7$.

Let $\mathcal{V}$ be the sequence of relations returned by FullRkRL. Note that we have $\|\mathbf{w}\|_\infty < \log_2 \Delta$ for any $\mathbf{w}$ with $\phi_{\mathcal{F}}(\mathbf{w}) = \mathfrak{a}$ where $\mathfrak{a}$ is reduced, since the norm of reduced ideals is bounded by $\sqrt{\Delta}$. Thus the off-diagonal entries are bounded in absolute value by Δ whereas the diagonal entries are at least $2n\Delta - \log_2 \Delta$ in size. This proves that $A(\mathcal{V})$ is strictly diagonally dominant.

Now we turn to the analysis of time and space bounds for FullRkRL. The main loop is executed n times. In it, PotRel is called $S = O^\sim(L)$ times since $n < y$. It executes in polynomial time. By Theorem 7.2 of [Ber02], step 5 executes in time which is a product of factors dominated by a polynomial in $\log \Delta$, and a factor of size L. The inner loop is executed at most S times. As can easily be seen, each of its iterations takes polynomial time. This concludes the proof of the running time bound given in the algorithm.

Each potential relation stored by FullRkRL requires polynomial space since $\mathbf{v}_j \in \iota(\mathbb{Z}^k)$ and $k = \operatorname{card} \mathcal{G}$ is polynomial in $\log \Delta$, and $\mathfrak{a}_j$ is reduced so that its standard representation takes space $O(\log \Delta)$. If $\Delta > 0$ we also have to store the relative generators of each potential relation. They require polynomial space again due to Lemma 4.26.

Finally, step 5 needs $O^\sim(L)$ space. Its space requirement is dominated by the storage of the product tree of the set $\{ \mathrm{N}(\mathfrak{a}_j) \mid j = 1, \ldots, S \}$, where S is by definition $O^\sim(L)$, and $\log_2(\mathrm{N}(\mathfrak{a}_j)) < \log \Delta$, since the $\mathfrak{a}_j$ are reduced. The size of the product tree is hence bounded by $O((\log_2 S) \cdot S \cdot \log_2 \Delta)$. This concludes the proof of the proposition. ∎

Lemma 4.29 *If* FullRkRL *returns successfully the set* $\mathcal{V}$ *of relations, then* $\det A(\mathcal{V}) < n^{n/2} \cdot (2n|\Delta|)^n$*.*

Proof This follows directly from the Hadamard bound. ∎

4.9 Generating essentially full relation lattices

In this section we will show how many times we need to call PotRel in order for the returned relation vectors to generate an essentially full relation lattice. Recall from 4.1 that a lattice $\mathcal{M} \subset \mathcal{L}_\mathcal{F}$ is called essentially full with respect to $\mathcal{G} \subset \mathcal{F}$ if $\iota(\mathcal{L}_\mathcal{G}) \subset \mathcal{M}$.

We need to study the probability that a potential relation generated by PotRel is smooth and the resulting relation lies outside some sub-lattice of $\mathcal{L}_\mathcal{F}$ that is not essentially full. We will treat the cases $\Delta < 0$ and $\Delta > 0$ separately.

If $\Delta < 0$ both properties depend deterministically on the exponent vector randomly drawn in the first step of PotRelIQ, i.e. the first component of the potential relation.

Denote the set of all y-smooth reduced ideals by $\mathcal{R}_y$, and its pre-image under $\phi_\mathcal{F}$ by V_y.

Lemma 4.30 *A potential relation* $(\mathbf{w}, \mathfrak{a})$ *is smooth if and only if* $\mathbf{w} \in V_y + \mathcal{L}_\mathcal{F}$.

Proof Let $(\mathbf{w}, \mathfrak{a})$ be a potential relation. Assume $\mathbf{w} \in \mathbf{v} + \mathcal{L}_\mathcal{F}$ with $\mathbf{v} \in V_y$. Then $\phi_\mathcal{F}(\mathbf{w}) \sim \phi_\mathcal{F}(\mathbf{v})$, where the latter ideal is reduced. Hence $\mathfrak{a} = \rho_0(\phi_\mathcal{F}(\mathbf{w}))$ equals $\phi_\mathcal{F}(\mathbf{v})$ or $\phi_\mathcal{F}(\mathbf{v})^\sigma$, and is, hence, smooth.

Let $(\mathbf{w}, \mathfrak{a})$ be a smooth potential relation. Then $\mathfrak{a} = \phi_\mathcal{F}(\mathbf{v})$ for some $\mathbf{v} \in \mathbb{Z}^n$ which—since $\mathfrak{a}$ is reduced—is thus a member of V_y. Since $\bar{\phi}_\mathcal{F}(\mathbf{w}) = \bar{\mathfrak{a}} = \bar{\phi}_\mathcal{F}(\mathbf{v})$, we have $\mathbf{w} - \mathbf{v} \in \mathcal{L}_\mathcal{F}$. Hence $\mathbf{w} \in \mathbf{v} + \mathcal{L}_\mathcal{F} \subset V_y + \mathcal{L}_\mathcal{F}$. ∎

Define the map

$$\kappa : V_y + \mathcal{L}_\mathcal{F} \longrightarrow \mathcal{L}_\mathcal{F} : \mathbf{w} \longmapsto \mathbf{w} - \mathbf{v} \quad \text{if } \rho_0(\phi_\mathcal{F}(\mathbf{w})) = \phi_\mathcal{F}(\mathbf{v})$$

A smooth potential relation $(\mathbf{w}, \mathfrak{a})$ yields the relation vector $\kappa(\mathbf{w})$. The conjugation operator is carried over from I_Δ to $\mathbb{Z}^n$ by $\phi_\mathcal{F}$, i.e., by the relation $\phi_\mathcal{F}(\mathbf{v})^\sigma = \phi_\mathcal{F}(\mathbf{v}^\sigma)$. A relation $\mathbf{w} \in \mathcal{L}_\mathcal{F}$ is called ambiguous if it can be written a $\mathbf{w} = \mathbf{v} - \mathbf{v}^\sigma$ for some $\mathbf{v} \in V_y$.

Lemma 4.31 *Let* $\mathcal{M} \subset \mathcal{L}_\mathcal{F}$ *be some relation lattice containing all ambiguous relations. A potential relation* $(\mathbf{w}, \mathfrak{a})$ *is smooth and yields a relation outside* $\mathcal{M}$ *if and only if* $\mathbf{w} \in V_y + \mathcal{L}_\mathcal{F} \setminus \mathcal{M}$.

Proof Assume the potential relation $(\mathbf{w}, \mathfrak{a})$ is smooth and $\kappa(\mathbf{w}) \in \mathcal{L}_\mathcal{F} \setminus \mathcal{M}$. Then $\mathbf{v} = \mathbf{w} - \kappa(\mathbf{w}) \in V_y$, and $\mathbf{w} = \mathbf{v} + \kappa(\mathbf{w}) \in V_y + \mathcal{L}_\mathcal{F} \setminus \mathcal{M}$.

On the contrary, if $\mathbf{w} = \mathbf{v} + \mathbf{u}$ with $\mathbf{v} \in V_y$ and $\mathbf{u} \in \mathcal{L}_\mathcal{F} \setminus \mathcal{M}$, then $\kappa(\mathbf{w})$ equals either $\mathbf{w} - \mathbf{v} = \mathbf{u} \in \mathcal{L}_\mathcal{F} \setminus \mathcal{M}$, or, if $\phi_\mathcal{F}(\mathbf{v})$ is ambiguous and $\rho_0(\mathbf{w}) = \mathbf{v}^\sigma$, it equals $\mathbf{w} - \mathbf{v}^\sigma = \mathbf{u} + \mathbf{v} - \mathbf{v}^\sigma$ which is also in $\mathcal{L} \setminus \mathcal{M}$. ∎

Proposition 4.32 *There exist explicitly computable constants $c_5 > 2$ and $\Delta_5 < 0$ with the following property.*

Let $\Delta < \Delta_5$. Suppose that $\mathcal{M} \subset \mathcal{L}_{\mathcal{F}}$ is a lattice which is not essentially full and contains all ambiguous relations. Then the probability that PotReliQ *returns a potential relation $(\mathbf{w}, \mathfrak{a})$ such that $\kappa(\mathbf{w}) \in \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ conditional on $\mathfrak{a}$ being smooth exceeds $1/c_5$.*

Proof Lemma 4.31 tells us that we need to establish the cardinality of $\iota(B_k(|\Delta|)) \cap (V_y + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M})$. Denote by $\bar{V_y}$ a subset of V_y that contains exactly one representative for each equivalence class in V_y with respect to conjugation.

The sets $\iota(B_k(|\Delta|)) \cap (V_y + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M})$ and $\iota(B_k(|\Delta|)) \cap (V_y + \mathcal{L}_{\mathcal{F}})$ split into the disjoint unions

$$\bigcup_{\mathbf{v} \in \bar{V_y}} \iota(B_k(|\Delta|)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}) \quad \text{and} \quad \bigcup_{\mathbf{v} \in \bar{V_y}} \iota(B_k(|\Delta|)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}}).$$

Hence it suffices to analyze the quotient between the cardinalities of the sets $\iota(B_k(|\Delta|)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M})$ and $\iota(B_k(|\Delta|)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}})$ for fix $\mathbf{v} \in V_y$.

Consider $(\mathbf{v} + \mathcal{L}_{\mathcal{F}}) \cap \iota(B_k(|\Delta| - h))$. We estimate its cardinality. Choose $\mathbf{v}' \in \mathbb{Z}^k$ such that $\mathbf{v} - \iota(\mathbf{v}') \in \mathcal{L}_{\mathcal{F}}$. This exists by Lemma 4.1 since $\mathcal{G}$ is a generating set for Cl_Δ. Then

$$\begin{aligned}
&\operatorname{card}\big((\mathbf{v} + \mathcal{L}_{\mathcal{F}}) \cap \iota(B_k(|\Delta| - h))\big) \\
&\qquad = \operatorname{card}\big((\iota(\mathbf{v}') + \mathcal{L}_{\mathcal{F}}) \cap \iota(B_k(|\Delta| - h))\big) \\
&\qquad = \operatorname{card}\big((\iota(\mathbf{v}') + \iota(\mathcal{L}_{\mathcal{G}})) \cap \iota(B_k(|\Delta| - h))\big) \\
&\qquad = \operatorname{card}\big(\mathbf{v}' + \mathcal{L}_{\mathcal{G}}) \cap B_k(|\Delta| - h)\big) \\
&\qquad \geq \frac{(|\Delta| - h)^k}{h} \cdot (1 - \frac{h-1}{|\Delta| - h}),
\end{aligned}$$

by Lemma 4.15.

Thus either $\operatorname{card}\big((\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}) \cap \iota(B_k(|\Delta| - h))\big)$ or $\operatorname{card}\big((\mathbf{v} + \mathcal{M}) \cap \iota(B_k(|\Delta| - h))\big)$ exceeds $(|\Delta| - h)^k/(2h) \cdot (1 - (h-1)/(|\Delta| - h))$.

Assume only the cardinality of the second one does. Choose $\mathbf{u} \in (\mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}) \cap \iota(\mathcal{L}_{\mathcal{G}})$ with $\|\mathbf{u}\|_\infty \leq h$. This is possible since $\mathcal{M}$ is not essentially full and $\det \mathcal{L}_{\mathcal{G}} = h$. Then shifting by u maps each element in $\mathcal{M} \cap \iota(B_k(|\Delta| - h))$ into $\mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M} \cap \iota(B_k(|\Delta|))$.

Thus we have proved that in either case

$$\big((\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}) \cap \iota(B_k(|\Delta|))\big) > \frac{(|\Delta| - h)^k}{2h} (1 - \frac{h-1}{|\Delta| - h}).$$

From Lemma 4.15 we also have

$$\operatorname{card}\big((\mathbf{v} + \mathcal{L}_{\mathcal{F}}) \cap \iota(B_k(|\Delta|))\big) < \frac{|\Delta|^k}{h} (1 + \frac{h-1}{|\Delta|})$$

In summary, the probability that PotRelIQ returns a potential relation $(\mathbf{w}, \mathfrak{a})$ with $\kappa(\mathbf{w}) \in \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ conditional on $\mathfrak{a}$ being smooth is bounded from below by

$$\frac{(|\Delta| - h)^{k-1}}{2|\Delta|^{k-1}} \cdot \frac{|\Delta| - 2h + 1}{|\Delta| - h - 1}$$

which is easily seen to be bounded from below by $\Omega(1)$ due to (4.7) and (4.8). ∎

Corollary 4.33 *Keeping the notations of Proposition 4.32, the probability that a call to* PotRelIQ *returns a smooth potential relation* $(\mathbf{w}, \mathfrak{a})$ *such that* $\kappa(\mathbf{w}) \in \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ *exceeds* $1/(c_5 \cdot L)$*, where* $L = L_{|\Delta|}(\frac{1}{2}, (1 + c_3(z))/(4z))$.

Proof Combine the probabilities from Propositions 4.27 and 4.32. ∎

In the real-quadratic situation we aim first to obtain a sub-lattice $\tilde{\mathcal{M}}$ of $\tilde{\mathcal{L}}_{\mathcal{F}}$ whose integral part $\mathcal{M}$ is essentially full. The analysis of PotRelRQ is exactly the same as that of PotRelIQ with the exception of an additional factor with size $O(\log \Delta)$ inherited from the deviation of $\mathfrak{r}$ from uniformity described by (4.14).

Lemma 4.34 *For a smooth potential extended relation* $(\mathbf{w}, \mathfrak{a}, \gamma)$ *we have* $\mathbf{w} \in V_y + \mathcal{L}_{\mathcal{F}}$*, and* $\mathbf{w} - \phi_{\mathcal{F}}^{-1}(\mathfrak{a}) \in \mathcal{L}_{\mathcal{F}}$.

Proof This is obvious since $\phi_{\mathcal{F}}(\mathbf{w}) \sim \mathfrak{a} \in I_{\mathcal{F}}$. ∎

Define the map

$$\kappa : \bigcup_{\mathbf{v} \in V_y} (\mathbf{v} + \mathcal{L}_{\mathcal{F}}) \times \{\mathbf{v}\} \longrightarrow \mathcal{L}_{\mathcal{F}} \,:\, (\mathbf{w}, \mathbf{v}) \longmapsto \mathbf{w} - \mathbf{v}$$

A smooth potential relation $(\mathbf{w}, \mathfrak{a}, \alpha)$ yields the relation vector $\kappa(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a}))$.

Lemma 4.35 *Let* $\mathcal{M} \subseteq \mathcal{L}_{\mathcal{F}}$ *be some relation lattice. For a smooth potential extended relation* $(\mathbf{w}, \mathfrak{a}, \alpha)$*, the corresponding relation vector* $\kappa(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a}))$ *lies in* $\mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ *if and only if* $\mathbf{w} \in \phi_{\mathcal{F}}^{-1}(\mathfrak{a}) + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$. ∎

We adapt Proposition 4.32 to the case $\Delta > 0$.

Proposition 4.36 *There exist integral* $\Delta_6 > 0$ *and a function* $c_6(\Delta) > 2$ *which grows at most linearly in* $\log \Delta$*, both explicitly computable, with the following property.*

Let $\Delta > \Delta_6$*. Suppose that* $\mathcal{M} \subset \mathcal{L}_{\mathcal{F}}$ *is a lattice which is not essentially full. Then the probability that* PotRelRQ *returns an extended potential relation* $(\mathbf{w}, \mathfrak{a}, \alpha)$ *such that* $\kappa(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a})) \in \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ *conditional on* $\mathfrak{a}$ *being smooth exceeds* $1/c_6(\Delta)$.

PROOF Due to Lemmata 4.34 and 4.35, the set of all pairs $(\mathbf{w}, \mathfrak{a})$ occurring in a smooth extended potential relation $(\mathbf{w}, \mathfrak{a}, \alpha)$ returned by POTRELRQ and such that $\kappa(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a})) \in \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ is the disjoint union

$$\bigcup_{\mathbf{v} \in V_y} \big(\iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M})\big) \times \{\mathbf{v}\}.$$

Likewise the set of all pairs $(\mathbf{w}, \mathfrak{a})$ occurring in a smooth extended potential relation $(\mathbf{w}, \mathfrak{a}, \alpha)$ returned by POTRELRQ without restriction is the disjoint union

$$\bigcup_{\mathbf{v} \in V_y} \big(\iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}})\big) \times \{\mathbf{v}\}.$$

Fix $\mathbf{v} \in V_y$. We analyze the probability that POTRELRQ returns an extended potential relation $(\mathbf{w}, \mathfrak{a}, \alpha)$ for which $(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a}))$ lies in $(\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}) \times \{\mathbf{v}\}$. It is bounded from below by

$$\frac{\operatorname{card}\big(\iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M})\big)}{\Delta^k} \times \min(\, \operatorname{Prob}(\mathfrak{r}(\rho_0(\phi_{\mathcal{F}}(\mathbf{w}))) = \mathbf{v}) \mid \mathbf{w} \in \iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}})\,). \quad (4.18)$$

By the same argument as in the proof of Proposition 4.32 we have

$$\frac{\operatorname{card}\big(\iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M})\big)}{\Delta^k} > \frac{(\Delta - h)^k}{2h\Delta^k}\,(1 - \frac{h-1}{\Delta - h}).$$

Moreover, by (4.14) the second term of (4.18) is bounded from below by $\log(2)/(2R) - 1/\Delta$.

The probability that POTRELRQ returns a smooth extended potential relation $(\mathbf{w}, \mathfrak{a}, \alpha)$ for which $(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a}))$ lies in $(\mathbf{v} + \mathcal{L}_{\mathcal{F}}) \times \{\mathbf{v}\}$ is bounded from above by

$$\frac{\operatorname{card}\big(\iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}})\big)}{\Delta^k} \times \max(\, \operatorname{Prob}(\mathfrak{r}(\rho_0(\phi_{\mathcal{F}}(\mathbf{w}))) = \mathbf{v}) \mid \mathbf{w} \in \iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}})\,). \quad (4.19)$$

Applying Lemma 4.15 we have

$$\frac{\operatorname{card}\big(\iota(B_k(\Delta)) \cap (\mathbf{v} + \mathcal{L}_{\mathcal{F}})\big)}{\Delta^k} < \frac{1}{h} \cdot (1 + \frac{h-1}{\Delta}).$$

Moreover, again by (4.14) the second term of (4.19) is bounded from above by $\log \Delta/(2R) + 1/\Delta$.

Taking all the bounds together we find that the sought conditional probability is bounded from below by

$$\frac{(\Delta - h)^{k-1}}{2\Delta^{k-1}}\,\frac{\Delta - 2h + 1}{\Delta + h - 1}\,\frac{\Delta \log 2 - 2R}{\Delta \log \Delta + 2R}.$$

This fraction is easily seen to be $\Omega(1/(1 + \log \Delta))$ and smaller than $1/2$. ∎

Corollary 4.37 *Keeping the notation of Proposition 4.36, the probability that a call to* PotRelRQ *returns a smooth extended potential relation* $(\mathbf{w}, \mathfrak{a}, \alpha)$ *such that* $\kappa(\mathbf{w}, \phi_{\mathcal{F}}^{-1}(\mathfrak{a})) \in \mathcal{L}_{\mathcal{F}} \smallsetminus \mathcal{M}$ *exceeds* $1/(c_6(\Delta) \cdot L)$*, where* $L = L_{|\Delta|}(\frac{1}{2}, (1 + c_3(z))/(4z))$*.*

Proof Combine the probabilities from Propositions 4.27 and 4.36. ∎

Lemma 4.38 *Let* $N \in \mathbb{Z}$ *be given, with* $N > 5$*, and* $M = \lceil \log_2 N \rceil$*. Let further* $L = \lceil L_{|\Delta|}(\frac{1}{2}, (1 + c_3(\Delta))/(4z)) \rceil$*. Assume* $\mathcal{V}$ *has been obtained from* $cNML$ *calls to* PotRel*, where* $c \in \mathbb{Z}$ *and* $c \geq c_5$*, if* $\Delta < 0$*, and* $c \geq c_6(\Delta)$*, if* $\Delta > 0$*. If* $\Delta < \Delta_5$ *or* $\Delta > \Delta_6$ *and* $\mathcal{M}$ *is a full-rank lattice in* $\mathcal{L}_{\mathcal{F}}$ *with determinant bounded by* 2^N *and containing all ambiguous relations if* $\Delta < 0$*, then* $\mathcal{M} + \langle \mathcal{V} \rangle$ *is with probability larger than* $\frac{1}{3}$ *essentially full.*

Proof Divide the potential relations into N blocks of cML each. Then the probability that no potential relation in a block is smooth and extends the relation lattice obtained up to this point provided the latter is not yet essentially full is smaller than

$$(1 - 1/(cL))^{cML} < 2^{-M} < 1/N.$$

The probability for all blocks to either find the relation lattice essentially full at entry or to contain at least one smooth potential relation extending the relation lattice is thus at least

$$(1 - 1/N)^N > 1/3$$

for $N \geq 6$. Since $\det \mathcal{M} \leq 2^N$ no strictly increasing sequence of lattices containing $\mathcal{M}$ can contain more than N elements. It follows that $\mathcal{M} + \langle \mathcal{V} \rangle$ is with probability larger than $\frac{1}{3}$ essentially full. ∎

Lemma 4.39 *Let* $N \in \mathbb{Z}$ *be given, with* $N > 5$*, and* $M = \lceil \log_2 N \rceil$*. Assume a sequence* $\mathcal{V}$ *has been obtained from calls to* PotRel *with input* $\Delta < \Delta_5$ *or* $\Delta > \Delta_6$*. Assume further that* $\mathcal{V}$ *contains more than* cNM *smooth relations, where* $c \in \mathbb{Z}$ *and* $c \geq c_5$*, if* $\Delta < 0$*, and* $c \geq c_6(\Delta)$*, if* $\Delta > 0$*. Let* $\mathcal{W}$ *contain the relations corresponding to the first* cNM *smooth relations in* $\mathcal{V}$*.*

If $\mathcal{M}$ *is a full-rank lattice in* $\mathcal{L}_{\mathcal{F}}$ *with determinant bounded by* 2^N*, then* $\mathcal{M} + \langle \mathcal{W} \rangle$ *is with probability larger than* $\frac{1}{3}$ *essentially full.*

Proof The proof is analogous to that of Lemma 4.38. ∎

For $\Delta < 0$, we are now in the position to formulate algorithm EssRlIQ that with large likelihood produces a sequence of relations which taken together generate an essentially full relation lattice. A listing of the algorithm can be found on the following page. The algorithm for $\Delta > 0$ will be presented in the next section where it will be combined with the computation of the regulator of the order.

The relations are generated by sub-algorithms FullRkRL, AmbRls and RelGenIQ. The listing for the first can be found on page 69, for the last on the facing page. We do not give a listing for AmbRls which computes all ambiguous relations in $\mathcal{L}_{\mathcal{F}}$. It is, however, an easy exercise to obtain AmbRls form Theorem 2.5 in [LP92]. See also [Cox89]. The essential determinant of the relation lattice is computed and printed in order to facilitate a check of the correctness of the result.

Algorithm 4.10: Generation of an ess. full relation lattice (IQ)

Input: Discriminant Δ, set $\mathcal{G}$ of reduced ideals

Output: A factor base $\mathcal{F}$, a sequence $(\mathbf{v}_i)$ of relations in $\mathcal{L}_{\mathcal{F}}$, and $\det\langle \mathbf{v}_i \rangle$: or Failure

---

EssRlIQ$(\Delta, \mathcal{G})$

1. Let $z \leftarrow 1/\sqrt{8}$ and $y = L_\Delta(\frac{1}{2}, z)$.
2. Enumerate factor base $\mathcal{F} \leftarrow \mathcal{F}_y \cup \mathcal{G}$, and let $n \leftarrow \operatorname{card} \mathcal{F}$.
3. $\mathcal{U} \leftarrow$ FullRkRL$(\Delta, \mathcal{F}, \mathcal{G})$.
4. **if** FullRkRL did not succeed **then print** Failure and **exit**
5. $\mathcal{V} \leftarrow$ AmbRls(Δ).
6. Let $N \leftarrow n \cdot \lceil \log_2(2n^{3/2}|\Delta|) \rceil$ and $M \leftarrow 1 + \lceil \log_2 N \rceil$.
7. Let $L \leftarrow \lceil L_{|\Delta|}(\frac{1}{2}, (1 + c_3(\Delta))/(4z)) \rceil$ and $S = ML\lceil c_5 \rceil$
8. $\mathcal{W} \leftarrow$ RelGenIQ$(\Delta, \mathcal{F}, \mathcal{G}, N, S)$.
9. If necessary, truncate $\mathcal{W}$ such that $m = \operatorname{card} \mathcal{W} < NM\lceil c_5 \rceil$.
10. Compute the lattice determinant $h \leftarrow$ EssDet$(A(\mathcal{V} \cup \mathcal{W}))$ repeating, if necessary, EssDet until it succeeds.
11. **print** $\mathcal{F}$, $\mathcal{U} \cup \mathcal{V} \cup \mathcal{W}$, and h.

Proposition 4.40 (GRH) *Let $z = 1/\sqrt{8}$, and fix some $c > 0$. For any $o > 0$, there exists an explicitly computable $\Delta_7 < 0$ with the following property.*

Given a fundamental discriminant $\Delta < \Delta_7$ and a set $\mathcal{G}$ of reduced ideals such that $\bar{\phi}_{\mathcal{F}}(\mathcal{G}) = \mathrm{Cl}_\Delta$ and $\operatorname{card} \mathcal{G} \le c \log^2|\Delta|$, Algorithm EssRlIQ *prints with probability bounded from below by $\frac{1}{3}$,*

- a set $\mathcal{V}$ of relations in $\mathcal{L}_{\mathcal{F}}$, with no more than $L_{|\Delta|}(\frac{1}{2}, z + o)$ elements, and
- the determinant of the lattice generated by $\mathcal{V}$.

Otherwise it announces failure.

With probability bounded from below by $\frac{1}{9}$, EssRlIQ *succeeds and*

- $\mathcal{V}$ generates an essentially full relation lattice, and
- h is the class number of the order $\mathcal{O}$ with discriminant Δ,

Algorithm 4.11: Generation of relation sequence (IQ)

Input: Discriminant Δ; generating set $\mathcal{G}$;
factor base $\mathcal{F} = \mathcal{F}_y$ with $y = L_{|\Delta|}(\frac{1}{2}, z)$ and $\operatorname{card}\mathcal{F} = n$;
number of blocks N, block length S

Output: Sequence $(\mathbf{v}_i)$ of relations in $\mathcal{L}_\mathcal{F}$

RelGenIQ$(\Delta, \mathcal{F}, \mathcal{G}, N, S)$
1. **for** $j = 1$ **to** $N \cdot S$
2. $\quad(\mathbf{w}_j, \mathfrak{a}_j) =$ PotRelIQ$(\Delta, \mathcal{F}, \mathcal{G}, \mathbf{0})$.
3. $\mathcal{N}_j \leftarrow \{\, p \in \mathcal{P}_y \mid p \mid \mathrm{N}(\mathfrak{a}_j) \,\}$ for $j = 1, \ldots, N \cdot S$
(computed using Algorithm 7.1 from [Ber02]).
4. **for** $j \leftarrow 1$ **to** $N \cdot S$
5. $\quad$Initialize $q \leftarrow 1$.
6. $\quad$**foreach** $p \in \mathcal{N}_j$
7. $\quad\quad$Find e_p such that $p^{e_p} \parallel \mathrm{N}(\mathfrak{a}_j)$.
8. $\quad\quad$$q \leftarrow q \cdot p^{e_p}$.
9. $\quad$**if** $q = \mathrm{N}(\mathfrak{a}_j)$ i.e. $\mathfrak{a}_j$ is y-smooth
10. $\quad\quad$Initialize $\mathbf{v} \leftarrow \mathbf{0} \in \mathbb{Z}^n$.
11. $\quad\quad$**foreach** $p \in \mathcal{N}_j$
12. $\quad\quad\quad$Find $\mathfrak{p}$ with $N(\mathfrak{p}) = p$ such that $\mathfrak{p}\mathfrak{a}_j$ is primitive.
13. $\quad\quad\quad$Set $\mathbf{v} = \mathbf{v} + e_p \cdot \phi_\mathcal{F}^{-1}(\mathfrak{p})$.
14. $\quad\quad$**print** $\mathbf{w}_j - \mathbf{v}$.

Time and space required for the execution of EssRlIQ *are bounded by* $L_{|\Delta|}(\frac{1}{2}, 3z + o)$

Proof We use notations and definitions as in the algorithm.

FullRkRL executed in step 3 of EssRlIQ succeeds with probability exceeding a $\frac{1}{3}$ and executes in time $L_{\Delta}(\frac{1}{2}, z + 1/(4z) + o(1))$ due to Proposition 4.28.

Assume FullRkRL succeeded. Then the probability that the lattice $\langle \mathcal{U} \cup \mathcal{V} \cup \mathcal{W} \rangle$ is essentially full exceeds $\frac{1}{3}$ due to Lemmata 4.38 and 4.39. This leads to a total success probability for EssRlIQ of $\frac{1}{9}$.

The analysis of time and space requirements of EssRlIQ up to step 10 are analogous to that of FullRkRL.

The run-time of AmbRls is dominated by the time needed to factor Δ. By the main result of [LP92], factoring can be done in time $L_{|\Delta|}(\frac{1}{2}, 1+o(1))$.

Each iteration of the loops in RelGenIQ executes in polynomial time. Each potential relation stored requires polynomial size.

The size of the set $\mathcal{N} = \{ N(\mathfrak{a}_j) \mid j = 1, \ldots, N \cdot S \}$ of numbers to be factored by Bernstein's algorithm equals the number of rounds in both loops and is the product of three terms: one dominated by a polynomial in $\log \Delta$, the second being $n < L(\frac{1}{2}, z + o(1)) + \operatorname{card} \mathcal{G}$, and the third equal to $L_{|\Delta|}(\frac{1}{2}, 1/(4z)+o(1))$. Each number to be factored is smaller than $\sqrt{|\Delta|}$. The run-time of Bernstein's algorithm in step 3 is hence bounded by $L_{|\Delta|}(\frac{1}{2}, z + 1/(4z) + o(1))$.

EssDet will be analyzed in Chapter 5. The running time of one call to EssDet is bounded by $O^{\sim}(nm^2) = L_{|\Delta|}(3z+o(1))$. Each call succeeds with probability larger than $\frac{1}{2}$. The expected time for completion of step 10 is hence double that of one call.

Combining the run-time bounds we see that the execution of EssRlIQ requires less than $L_{|\Delta|}(\frac{1}{2}, \max(3z, z + 1/(4z)) + o(1))$ time. The maximum is minimized when $z = 1/\sqrt{8}$ at $L_{|\Delta|}(\frac{1}{2}, 3/\sqrt{8})$. ∎

Note 4.41 The factorization in AmbRls may use RelGenIQ (or possibly several calls to RelGenIQ with varying parameters) for relation generation.

It is possible to avoid using procedure AmbRls at the cost of at most doubling the effort in subsequent steps by randomizing among the conjugated reduced ideals in an ambiguous class whenever this occurs in a call to PotRelIQ.

We will recall in Section 4.11 how to check the correctness of the class number returned by EssRlIQ. In Section 4.12 we will give a class group structure algorithm and an algorithm for the solution of the discrete logarithm problem in Cl_{Δ} on the basis of EssRlIQ.

4.10 Regulator computation

In this section, we will give a probabilistic algorithm for the computation of class number and regulator of a maximal real quadratic order that also runs in expected time $L_\Delta(\frac{1}{2}, 3/\sqrt{8} + o(1))$.

Assume we are given a set $\mathcal{V}$ of extended relations of cardinality m. Let $\tilde{\mathcal{M}}$ be the extended relation lattice generated by $\mathcal{V}$. Then elements in $\tilde{\mathcal{M}} \cap \ker \pi$ correspond to units of $\mathcal{O}$.

Assume, moreover,

$$\pi(v) \in \big\langle \pi(\mathcal{V} \smallsetminus \{v\}) \big\rangle. \tag{4.20}$$

Order $\mathcal{V}$ such that $v > w$ for all $w \in \mathcal{V} \smallsetminus \{v\}$. Solve

$$A\big(\pi(\mathcal{V} \smallsetminus \{v\})\big)\, \mathbf{x} = \pi(v). \tag{4.21}$$

Then for the solution $\mathbf{x} \in \mathbb{Z}^{\mathcal{V} \smallsetminus \{v\}}$, the extended relation

$$-v + \sum_{w \in \mathcal{V} \smallsetminus \{v\}} \mathbf{x}_w \cdot w$$

is in the kernel of π and corresponds to the unit

$$\epsilon = \varepsilon_\Delta^l = \tau(v)^{-1} \cdot \prod_{w \in \mathcal{V} \smallsetminus \{v\}} \tau(w)^{\mathbf{x}_\mathbf{w}}. \tag{4.22}$$

for some $l \in \mathbb{Z}$ where $\tau(u)$ denotes the positive generator of extended relation u.

We determine the probability that two extended relations drawn from a sufficiently large set satisfy (4.20).

Lemma 4.42 *Let $\mathcal{M} \subset \mathbb{Z}^n$ be a full-rank lattice with determinant smaller than 2^N for some $N \in \mathbb{Z}$. Let further $\mathcal{U}$ be a sequence of vectors $\mathbf{u} \in \mathbb{Z}^n$ with at least $2N$ elements. Then the probability that two randomly drawn $\mathbf{v}, \mathbf{w} \in \mathcal{U}$ satisfy $\mathbf{v}, \mathbf{w} \in \mathcal{M} + \big\langle \mathcal{U} \smallsetminus \{\mathbf{v}, \mathbf{w}\} \big\rangle$ is larger than $\frac{1}{4}$.*

Proof Let $\mathcal{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$. Let further $\mathcal{M}_j = \mathcal{M} + \big\langle \mathbf{u}_i \mid i \le j \big\rangle$ for $j = 0, \ldots, m$.

Since the determinant of $\mathcal{M}$ is smaller than 2^N we know that any strictly increasing sequence of lattices in $\mathbb{Z}^n$ that contain $\mathcal{M}$ has fewer than N elements. Hence there are fewer than N vectors $\mathbf{u}_i \in \mathcal{U}$ such that $\mathbf{u}_i \notin \mathcal{M}_{i-1}$.

Choose randomly two integers $a < b$ from the interval $[1, m]$. Since $m \ge 2N$ we find that with probability larger than $\frac{(m-N+1)(m-N)}{m(m-1)} > \frac{1}{4}$ we have $\mathbf{u}_a \in \mathcal{M}_{a-1}$ and $\mathbf{u}_b \in \mathcal{M}_{b-1}$.

Since $a < b$ we have immediately $\mathbf{u}_a \in \mathcal{M} + \big\langle \mathcal{U} \smallsetminus \{\mathbf{u}_a, \mathbf{u}_b\} \big\rangle$. We have also $\mathbf{u}_b \in \mathcal{M} + \big\langle \mathcal{U} \smallsetminus \{\mathbf{u}_a, \mathbf{u}_b\} \big\rangle$, since $\mathbf{u}_a \in \mathcal{M}_{a-1}$ implies $\mathcal{M}_{b-1} = \mathcal{M}_{a-1} + \big\langle \mathbf{u}_i \mid a < i < b \big\rangle$. ∎

The lattice $\mathcal{M}$ from Lemma 4.42 can be obtained by FullRkRL. Next, we determine the number of calls to PotRel that will yield sufficiently large $\mathcal{U}$.

Lemma 4.43 *Let $B \in \mathbb{Z}$ be given, with $B > 5$, and $M = \lceil \log_2 B \rceil$. Let further $L = \lceil L_{|\Delta|}(\frac{1}{2}, (1 + c_3(\Delta))/(4z)) \rceil$. Assume $\mathcal{V}$ has been obtained from BML calls to* PotRel. *Then $\mathcal{V}$ contains with probability larger than $\frac{1}{3}$ at least B smooth potential relations.*

Proof Divide the potential relations into B blocks of ML each. Then the probability that no potential relation in each block is smooth is

$$(1 - 1/L)^{ML} < 2^{-M} < 1/B.$$

The probability that in all blocks taken together there are at least B smooth potential relations is thus at least

$$(1 - 1/B)^B > 1/3$$

for $B \geq 6$. ∎

We have seen that FullRkRL followed by a sufficiently large number of calls to PotRel will yield enough potential relations to extract two units via (4.21) and (4.22), where $\mathcal{V}$ is the set of extended relations computed from the potential ones. We want to determine the probability that these two units generate the full unit group.

Two units ε_Δ^r and ε_Δ^s generate the full unit group (mod ± 1) if and only if $\gcd(r, s) = 1$. All else being equal ($\mathcal{V} \setminus \{v, w\}$, and the integral parts of v and w), the exponent r depends only on the number of times the cycles where traversed during the calls to RandRed in the computation of v and w. This number is coming with close to uniform distribution from an interval of width $\Delta \log \Delta / R$.

We analyze the probability that two exponents coming randomly from large intervals are co-prime.

Lemma 4.44 *Let $A, B, M \in \mathbb{Z}$ with $\log(|A - B| + 1) < M/100$. Consider the set $S = \{ (x, y) \in \mathbb{Z}^2 \mid A \leq x < A + M, B \leq y < B + M \}$. If $M \gg 0$ then there are more than $M^2/2$ pairs $(x, y) \in S$ with $\gcd(x, y) = 1$.*

Proof We define the following subsets of S:

$$T = \{ (x, y) \in S \mid \gcd(x, y) \neq 1 \},$$
$$T_p = \{ (x, y) \in S \mid p \mid \gcd(x, y) \}$$

where p denotes some prime number. We need to show that $\operatorname{card} T < M^2/2$. We will show instead that

$$\sum_{p \leq M} \operatorname{card} T_p + \operatorname{card} \bigcup_{p > M} T_p < M^2/2$$

which is certainly sufficient. Note that for any two $p, q > M$ with $p \neq q$ the sets T_p and T_q are disjoint.

Let $p \leq M$. Then a simple counting argument shows that $\operatorname{card} T_p < (1 + \lfloor M/p \rfloor)^2$. Thus

$$\begin{aligned}\sum_{p \leq M} \operatorname{card} T_p &< \sum_{p \leq M} (1 + M/p)^2 \\ &< M(\log\log M + O(1)) + M^2 P(2),\end{aligned}$$

where P is the prime zeta function, and $P(2) = 0.452....$

Let $p > M$. Then $\operatorname{card} T_p \leq 1$. For any $d \in \mathbb{Z}$ we define yet another set $U_d = \{\, (x, y) \in S \mid x - y = d \,\}$. If $T_p \cap U_d \neq \emptyset$ then $p \mid d$. Thus since $|d| < |A - B| + M$

$$\operatorname{card}(U_d \cap \bigcup_{p > M} T_p) < \log(|A - B| + M).$$

From this we deduce

$$\begin{aligned}\operatorname{card} \bigcup_{p > M} T_p &= \sum_{d = A - B - M}^{A - B + M} \operatorname{card}(U_d \cap \bigcup_{p > M} T_p) \\ &< 2M(\log(|A - B| + M)) < M^2/50 + M \log M.\end{aligned}$$

Adding the two estimates we obtain the desired result for sufficiently large M. ∎

Thus we are ready to formulate algorithm EssRlRQ which computes a generating set of an extended relation lattice. The integral part of this lattice is with constant probability essentially full.

EssRlRQ also computes the essential determinant h of the integral part of the lattice, and a real number R which is close to a regulator multiple. We will prove below that with constant probability h equals the class number, and R is a regulator approximation.

Under the assumption of the GRH, it is possible to verify whether h and R are correct. Thus we also have a check on the property that the extended relation lattice obtained has essentially full integral part.

The listing for EssRlRQ can be found on the next page, sub-algorithms FullRkRL on page 69, and RelGenRQ on page 83.

Proposition 4.45 (GRH) *Let $z = 1/\sqrt{8}$, and fix some $c > 0$. For any $o > 0$, there exists an explicitly computable $\Delta_8 > 0$ with the following property.*

Given a fundamental discriminant $\Delta > \Delta_8$ and a set $\mathcal{G}$ of reduced ideals such that $\bar{\phi}_{\mathcal{F}}(\mathcal{G}) = \mathrm{Cl}_\Delta$ and $\operatorname{card} \mathcal{G} < c \log^2 \Delta$, Algorithm EssRlRQ *prints with probability bounded from below by $1/36$,*

Algorithm 4.12: Generation of ess. full relation lattice (RQ)

Input: Discriminant Δ and a set $\mathcal{G}$ of reduced ideals

Output: Class number h_Δ, and $R \in \mathbb{Q}$ such that $|R - R_\Delta| < 1$

EssRlRQ$(\Delta, \mathcal{G})$

1. Let $z \leftarrow 1/\sqrt{8}$ and $y = L_\Delta(\frac{1}{2}, z)$.
2. Enumerate factor base $\mathcal{F} \leftarrow \mathcal{F}_y \cup \mathcal{G}$, and let $n \leftarrow \operatorname{card} \mathcal{F}$.
3. $\mathcal{V} = \big((\mathbf{v}_i, \alpha_j)\big) \leftarrow$ FullRkRL$(\Delta, \mathcal{F}, \mathcal{G})$.
4. **if** FullRkRL did not succeed **then print** Failure and **exit**
5. Let $N \leftarrow n \cdot \lceil \log_2(2n^{3/2}|\Delta|) \rceil$ and $M \leftarrow 1 + \lceil \log_2 N \rceil$.
6. Let $L \leftarrow \lceil L_{|\Delta|}(\frac{1}{2}, (1 + c_3(\Delta))/(4z)) \rceil$ and $S = ML\lceil c_6(\Delta) \rceil$
7. Compute relations $\mathcal{W} = \big((\mathbf{w}_j, \beta_j)\big)_{j=n+1}^{n+m} \leftarrow$ RelGenRQ$(\Delta, \mathcal{F}, \mathcal{G}, S, N)$.
8. Truncate $\mathcal{W}$, if necessary, such that $m = \operatorname{card} \mathcal{W} \le \lceil c_6(\Delta) \rceil NM$.
9. Compute the determinant $h \leftarrow$ EssDet$(A(\pi(\mathcal{V} \cup \mathcal{W})))$.
10. Repeat to call EssDet, if necessary, with the same input until it succeeds.
11. Let $m \leftarrow \operatorname{card} \mathcal{W}$, and choose randomly $a, b \in [1, m]$.
12. Let $(\mathbf{u}_1, \gamma_1) \leftarrow (\mathbf{w}_{n+a}, \beta_{n+a})$ and $(\mathbf{u}_2, \gamma_2) \leftarrow (\mathbf{w}_{n+b}, \beta_{n+b})$.
13. Let $\mathcal{W}' \leftarrow \mathcal{W} \setminus \{(\mathbf{w}_{n+a}, \beta_{n+a}), (\mathbf{w}_{n+b}, \beta_{n+b})\}$.
14. Let $\mathbf{x}_s \leftarrow$ DSolv$(A(\pi(\mathcal{V} \cup \mathcal{W}')), \mathbf{u}_s)$ for $s = 1, 2$.
15. **if** DSolv fails **then print** Failure and **exit**.
16. Let $\epsilon_s \leftarrow \gamma_s^{-1} \cdot \prod_i \alpha_i^{x_{s,i}} \cdot \prod_{j \neq a,b} \beta_j^{x_{s,j}}$ for $s = 1, 2$.
17. Compute ϵ by applying Algorithm `generating_unit` from [Mau00] to the set $\{\epsilon_1, \epsilon_2\}$.
18. Compute $R \leftarrow$ Log$(\epsilon, 1)$.
19. **print** $\mathcal{V} \cup \mathcal{W}$, h and R.

Algorithm 4.13: Generation of relation sequence (RQ)

Input: Discriminant Δ; factor base $\mathcal{F} = \mathcal{F}_y$ with $y = L_{|\Delta|}(\frac{1}{2}, z)$ and $\operatorname{card} \mathcal{F} = n$; generating set $\mathcal{G}$, number of blocks N, block length S

Output: Sequence $(\mathbf{v}_i)$ of relations in $\mathcal{L}_\mathcal{F}$

RelGenRQ$(\Delta, \mathcal{F}, \mathcal{G}, N, S)$

1. **for** $j = 1$ **to** $N \cdot S$
2. $\quad (\mathbf{w}_j, \mathfrak{a}_j, \alpha_j) =$ PotRelRQ$(\Delta, \mathcal{F}, \mathcal{G}, \mathbf{0})$.
3. $\mathcal{N}_j \leftarrow \{\, p \in \mathcal{P}_y \mid p \mid \mathrm{N}(\mathfrak{a}_j) \,\}$ for $j = 1, \ldots, N \cdot S$ (computed using Algorithm 7.1 from [Ber02]).
4. **for** $j \leftarrow 1$ **to** $N \cdot S$
5. $\quad$ Initialize $q \leftarrow 1$.
6. $\quad$ **foreach** $p \in \mathcal{N}_j$
7. $\quad\quad$ Find e_p such that $p^{e_p} \parallel \mathrm{N}(\mathfrak{a}_j)$.
8. $\quad\quad$ $q \leftarrow q \cdot p^{e_p}$.
9. $\quad$ **if** $q = \mathrm{N}(\mathfrak{a}_j)$ i.e. $\mathfrak{a}_j$ is y-smooth
10. $\quad\quad$ Initialize $\mathbf{v} \leftarrow \mathbf{0} \in \mathbb{Z}^n$.
11. $\quad\quad$ **foreach** $p \in \mathcal{N}$
12. $\quad\quad\quad$ Find $\mathfrak{p}$ with $N(\mathfrak{p}) = p$ such that $\mathfrak{p}\mathfrak{a}_j$ is primitive.
13. $\quad\quad\quad$ Set $\mathbf{v} = \mathbf{v} + e_p \cdot \phi_\mathcal{F}^{-1}(\mathfrak{p})$.
14. $\quad\quad$ **print** $\mathbf{w}_j - \mathbf{v}$ and α_j.

- *a set* $\mathcal{V}$ *of extended relations in* $\tilde{\mathcal{L}}_{\mathcal{F}}$*, where* $\mathcal{F} = \mathcal{F}_y \cup \mathcal{G}$*, with no more than* $L_\Delta(\frac{1}{2}, z + o)$ *elements,*
- *the determinant of the lattice generated by* $\pi(\mathcal{V})$*,*
- *an approximation* $R \in \mathbb{Q}$ *to a regulator multiple such that* $|R - tR_\Delta| < 1$*, for some* $t \in \mathbb{Z}$*.*

Otherwise it announces failure.

With probability bounded from below by $\Omega(1/\log^2 \Delta)$*,* EssRlRQ *succeeds and*

- $\pi(\mathcal{V})$ *generates a relation lattice which is essentially full relative to* $\mathcal{G}$*,*
- h *is the class number of the order* $\mathcal{O}$*,*
- $t = 1$*, i.e.,* R *is an approximation to the regulator itself.*

Time and space required for the execution of EssRlRQ *are bounded by* $L_\Delta(\frac{1}{2}, 3z + o)$*.*

Proof We use the notation introduced in the statement of the proposition and the algorithm listing.

We begin by bounding the execution time of EssRlRQ.

The computation of the factor base $\mathcal{F}$ takes time $O^\sim(n) = L_\Delta(\frac{1}{2}, z + o(1))$.

The call to FullRkRL takes time $L_\Delta(\frac{1}{2}, z + 1/(4z) + o(1))$ as proved in Proposition 4.28.

In RelGenRQ, PotRelRQ is called $NML\lceil c_6(\Delta)\rceil = L_\Delta(\frac{1}{2}, z+1/(4z)+o(1))$ times. Each call takes polynomial time due to Lemma 4.26. The same applies to the second loop in RelGenRQ.

The time for the factorization in step 3 of RelGenRQ is bounded by the product of an explicit term polynomial in $\log \Delta$, and the number of potential relations generated previously. This bound can, hence, again be written in the form $L_\Delta(\frac{1}{2}, z+1/(4z)+o(1))$ concluding the run-time analysis of RelGenRQ.

Before step 9 of EssRlRQ, we have $\operatorname{card} \mathcal{V} = n$ and $\operatorname{card} \mathcal{W} = m \leq 2NM = O^\sim(n)$. As proved in Chapter 5, EssDet in step 9 and DSolv in step 14 need time $O^\sim(nm^2)$. This can be written as $L_\Delta(\frac{1}{2}, 3z + o(1))$.

We turn to the run-time analysis of step 17. The generators α_i and β_j of all the relations obtained previously are given in power product representation. Tracing through the algorithm and using Lemma 4.17 and Lemma 4.22 we see that:

- the sum of exponents in this power product representation is bounded by $(k + 2)(\log_2 \Delta + 1)$ where $k = \operatorname{card} \mathcal{G}$;
- each factor is given in compact representation with no more than $O^\sim(\log \Delta)$ components;
- each component has height not exceeding $3\Delta^{9/4}$ and denominator not exceeding Δ.

The size of the entries of $\mathbf{x}_s$, where $s = 1, 2$ can be bounded due to Proposition 5.5 by $\log\|\mathbf{x}\| = O^{\sim}(n)$. Moreover, we know that $\operatorname{card}\mathcal{V} = n$, and $\operatorname{card}\mathcal{W}' = O^{\sim}(n)$ by construction. Hence, Lemma 2.23 tells us that it is possible to evaluate Log on ϵ_s as obtained in step 16 in time $O^{\sim}(n\cdot(\mu(n)-\log\delta))$ where δ denotes the required error bound.

We conclude that the Algorithm `rgcd_frac` of [Mau00] which requires only $\Omega(1)$ precision of its arguments and which dominates the run-time of `generating_unit` executes in time $O^{\sim}(n^3)$ if quadratic arithmetic is used, $O^{\sim}(n^2)$ if we apply pseudo-linear arithmetic.

Since all remaining steps of EssRlRQ can easily be seen to run in polynomial time, we get a total run-time bound of $L_{\Delta}(\frac{1}{2}, \max(3z, z + 1/(4z)) + o(1))$ which is minimized for $z = 1/\sqrt{8}$ at $L_{\Delta}(\frac{1}{2}, 3/\sqrt{8})$.

Next, we prove a lower bound for the success probability of EssRlRQ.

FullRkRL succeeds with probability at least $\frac{1}{3}$ due to Proposition 4.28. By Lemma 4.38, the probability that $\langle\pi(\mathcal{V}\cup\mathcal{W})\rangle$ is essentially full after step 7 is larger than $\frac{1}{3}$. If $\mathcal{W}$ is truncated in step 8, then $\langle\pi(\mathcal{W}\cup\mathcal{V})\rangle$ is still essentially full with probability larger than $\frac{1}{3}$ due to Lemma 4.39. Hence the determinant computed in step 9 equals h_{Δ} with total probability $\frac{1}{9}$.

Moreover, since $c_6(\Delta) > 2$ we are assured by Lemma 4.43 that with probability larger than $\frac{1}{3}$ the sequence $\mathcal{W}$ contains more than $2N$ relations. Hence by Lemma 4.42 the relations chosen in step 11 are with probability exceeding $\frac{1}{4}$ such that the Diophantine systems

$$A\big(\pi(\mathcal{V}\cup\mathcal{W}')\big)\,\mathbf{x}_s = \mathbf{u}_s,$$

$s = 1, 2$, are both solvable.

Finally, we estimate the probability that η_1 and η_2 generate the full unit group. It suffices to compute a lower bound for this to happen conditional on

- a fix pair of positions a and b having been chosen in step 11;
- a fixed set of potential relations having been computed at positions other than those corresponding to a and b;
- the potential relations at positions which correspond to a and b containing fixed pairs $(\mathbf{w}_s, \mathfrak{a}_s)$, $s = 1, 2$, of exponent vectors $\mathbf{w}_s \in V_y \cap \iota(B_k(\Delta))$ and reduced ideals $\mathfrak{a}_s \in V_y$;
- fix solution vectors $\mathbf{x}_s$ having been arrived at by DSolv in step 14.

All other data occurring in the algorithm except γ_1, γ_2, η_1, η_2, and of course ϵ and R depend in a deterministic fashion on these choices. This concerns in particular the expressions

$$E_s = \prod_i \alpha_i^{x_{s,i}} \cdot \prod_{j\neq a,b} \beta_j^{x_{s,j}}, \quad s = 1, 2.$$

We have already established that $\log|\operatorname{Log} E_s| = O^{\sim}(n)$. Hence we have for sufficiently large Δ that $\log|\operatorname{Log} E_1 - \operatorname{Log} E_2| < (\Delta\log\Delta)/(100R)$.

Moreover, the set of all γ_s that can occur under these constraints is contained in

$$\{\, \underline{\alpha}_s \varepsilon_\Delta^{e_s} \mid 0 \le e_s < M = \Delta \log \Delta / R \,\}, \quad s = 1, 2,$$

respectively, for some fix $\underline{\alpha}_1$ and $\underline{\alpha}_2$. It follows that the exponents l defined by (4.22) vary in intervals $[A, A+M]$, and $[B, B+M]$, respectively, with $\log|A-B| < M/100$. Hence there are by Lemma 4.44 at least $M^2/2$ pairs (e_1, e_2) leading to the unit ϵ computed in step 17 being fundamental.

Finally, the third statement of Proposition 4.23 shows that the probability that one of the good $M^2/2$ pairs is chosen is bounded from below by $\Omega(1/\log^2 \Delta)$. ∎

We will recall in Section 4.11 how to check the correctness of class number and regulator approximation returned by EssRlRQ. In Section 4.12 we will give a class group structure algorithm and algorithms for the solution of the principal ideal and extended discrete logarithm problems in $\mathcal{O}$ on the basis of EssRlRQ.

4.11 Checking correctness

In order to verify the correctness of the results we need to assure ourselves that class number and, if $\Delta > 0$, regulator printed by EssRlIQ or EssRlRQ are correct. We summarize the well-known method for this purpose.

Since all vectors output by these algorithms are indeed relations (extended ones if $\Delta > 0$) due to Lemma 4.26 we are assured that the lattice generated by the printed vectors has a determinant which is a multiple of the class number.

Moreover, ϵ computed in step 17 of EssRlRQ is certainly a unit. This implies that the rational number R printed is an approximation to a regulator multiple, i.e., there is some $t \in \mathbb{Z}$ such that

$$|R - t \cdot R_\Delta| < 1.$$

Assume we can find some $H \in \mathbb{Q}$ such that $H \le h_\Delta R_\Delta < \frac{3}{2}H$. If h is a non-trivial multiple of h_Δ, or $t > 1$, then $hR > \frac{7}{4}H$, which can be effectively tested.

In order to find such H we use the analytic class number formula (4.7), or (4.9), and the following proposition which bounds the error incurred by computing only a small number of terms in the Euler product for $L(1, \chi)$.

Proposition 4.46 (GRH) *Let χ be the quadratic character defined by $\chi(a) = \left(\frac{\Delta}{a}\right)$. Then there exist absolute and effectively computable constants c_1 and c_2 such that for all $x > c_1 \log^2 \Delta$*

$$\left| 1 - \prod_{p > x} \left(1 - \frac{\chi(p)}{p}\right) \right| < c_2 x^{-1/2} (\log|\Delta| + \log x)$$

PROOF See e.g. [Oes79]. ∎

Thus we can compute a truncated Euler product that approximates $L(1,\chi)$ sufficiently well in polynomial time.

Proposition 4.47 *There exists an algorithm that on input of the discriminant Δ of some order $\mathcal{O}$, and two numbers $h = rh_\Delta$, and $R \in \mathbb{Q}$ such that $|R - t \cdot R_\Delta| < 1$ with $r, t \in \mathbb{Z}$ decides in time polynomial in $\log|\Delta|$ whether $r = t = 1$ or not.* ∎

The algorithm whose existence is claimed in Proposition 4.47 will be called INVCHECK.

4.12 Summary

In this section, we will state the algorithms for the solution of the problems from Chapter 1, and their properties.

Theorem 4.48 (GRH) *There are two algorithms with the following properties:*

On input of any fundamental discriminant $\Delta \ll 0$, the first algorithm returns the class number and class group structure of the order with discriminant Δ with probability that independently of Δ exceeds $1/9$.

The second algorithm does the following on input of any fundamental discriminant $\Delta \ll 0$ and of two reduced ideals $\mathfrak{a}, \mathfrak{b} \in I_\Delta$: with probability exceeding $1/9$ independently of the input it either returns minimal n such that $\mathfrak{a} \sim \mathfrak{b}^n$ if it exists, or announces that $\mathfrak{a} \notin \langle [\mathfrak{b}] \rangle$ otherwise.

In the remaining cases both algorithms pronounce to have failed.

The expected run-time of both algorithms is bounded by $L_\Delta(\frac{1}{2}, 3/\sqrt{8} + \epsilon)$

PROOF We will prove that PCLIQ and PDLIQ whose listings can be found on the following page exhibit the properties claimed by the theorem. We begin by showing correctness and success probability of the algorithms.

Proposition 4.9 assures us that $\mathcal{G}$ is indeed a generating set for the class group Cl_Δ.

By Proposition 4.45 we know that with probability larger than $1/9$, the set $\mathcal{V}$ computed by ESSRLIQ generates an essentially full relation lattice. If ESSRLIQ does not fail, but returns $\mathcal{V}$ such that $\langle \mathcal{V} \rangle$ is not essentially full, then this is detected by INVCHECK, and FAILURE announced.

This settles success probability and correctness of PCLIQ, since the algorithms for HNF and SNF computation succeed unconditionally and are correct.

The correctness of PDLIQ follows finally from Corollary 4.8.

We turn to the analysis of the run-time required by the algorithms.

Algorithm 4.14: Class group structure (IQ)

Input: Δ fundamental discriminant

Output: class number h, and class group invariants s_i; or FAILURE

PCLIQ(Δ)
1. Let $c \leftarrow 3\log^2|\Delta|$.
2. Enumerate generating set $\mathcal{G} \leftarrow \mathcal{F}_c$ and let $l = \operatorname{card}\mathcal{G}$.
3. Compute relation vectors $(\mathcal{V}, h) \leftarrow \text{EssRlIQ}(\Delta, \mathcal{G})$.
4. **if** INVCHECK(Δ, h) fails **then print** FAILURE and **exit**.
5. Compute essential part of HNF: $H \leftarrow \text{EssHNF}(A(\mathcal{V}), h)$.
6. Compute SNF of essential part: $\text{diag}(s_1, \ldots, s_l) \leftarrow \text{SNF}(H)$.
7. **print** h and $s_1, \ldots, s_l$.

Algorithm 4.15: Discrete Logarithm (IQ)

Input: Δ fundamental discriminant, reduced ideals $\mathfrak{a}$, and $\mathfrak{b}$

Output: Minimal $k \in \mathbb{N}$ such that $\mathfrak{a} \sim \mathfrak{b}^k$; $\emptyset$ if no such k exists; or FAILURE

PDLIQ(Δ)
1. Let $c \leftarrow 3\log^2|\Delta|$.
2. Enumerate generating set $\mathcal{G} \leftarrow \{\mathfrak{b}, \mathfrak{a}\} \cup \mathcal{F}_c$ and let $l = \operatorname{card}\mathcal{G}$.
3. Compute relation vectors $(\mathcal{V}, h) \leftarrow \text{EssRlIQ}(\Delta, \mathcal{G})$.
4. **if** INVCHECK(Δ, h) fails **then print** FAILURE and **exit**.
5. Compute essential part of HNF: $H = (h_{i,j}) \leftarrow \text{EssHNF}(A(\mathcal{V}), h)$.
6. **if** $h_{2,2} = 1$ **then print** $h_{1,1} - h_{1,2}$,
7. **else print** $\emptyset$.

Proposition 4.9 assures us that $\mathcal{G}$ is indeed a generating set for the class group Cl_Δ. It can be generated in polynomial time.

By Proposition 4.40, the computation of the relation vectors takes time $L_{|\Delta|}(\frac{1}{2}, 3z + o(1))$ and produces a set $\mathcal{V}$ of relations with no more than $L_{|\Delta|}(\frac{1}{2}, 3z + o(1))$.

InvCheck takes polynomial time.

The computation of the essential part of the HNF of $A(\mathcal{V})$ is computed in time $O^{\sim}(nm^2) = L_{|\Delta|}(\frac{1}{2}, 3z + o(1))$ where n is the dimension of the relation lattice, and $m = \operatorname{card} \mathcal{V}$ the number of relations computed.

Finally, the computation of the invariants of Cl_Δ from the essential part H of the HNF of $A(\mathcal{V})$ takes polynomial time, since H is an $k \times k$ matrix and $k = 3\log^2|\Delta|$. ∎

Theorem 4.49 (GRH) *There are two algorithms with the following properties.*

On input of any fundamental discriminant $\Delta \gg 0$ the first algorithm returns the class number, class group structure and regulator of the order with discriminant Δ with probability that exceeds $\Omega(1/\log^2 \Delta)$.

The second algorithm does the following on input of any fundamental discriminant $\Delta \gg 0$ and of two reduced ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}_\Delta$: with probability exceeding $\Omega(1/\log^2 \Delta)$ independently of $\mathfrak{a}$ and $\mathfrak{b}$ it either returns minimal n and $\alpha \in \mathcal{K}$ in compact representation with $0 \leq \operatorname{Log} \alpha < R_\Delta$ such that $\mathfrak{a} = \alpha\mathfrak{b}^n$ if they exist, or announces that $\mathfrak{a} \notin \langle [\mathfrak{b}] \rangle$ otherwise.

In the remaining cases both algorithms pronounce to have failed.

The expected run-time of both algorithms is bounded by $L_\Delta(\frac{1}{2}, 3/\sqrt{8} + \epsilon)$.

Proof We will prove that PCLRQ and PDLRQ whose listings can be found on the next page exhibit the properties claimed by the theorem. We begin by showing correctness and success probability of the algorithms.

Proposition 4.9 assures us that $\mathcal{G}$ is indeed a generating set for the class group Cl_Δ.

By Proposition 4.45 we know that with probability larger than a bound in $\Omega(1/\log^2 \Delta)$, the set $\mathcal{V}$ computed by EssRlRQ generates an essentially full relation lattice, and $|R - R_\Delta| < 1$. If EssRlRQ does not fail, but returns $\mathcal{V}$ such that $\langle \pi(\mathcal{V}) \rangle$ is not essentially full, or R such that $|R - t \cdot R_\Delta| < 1$ with $t \neq 1$, then this is detected by InvCheck, and Failure announced.

This settles success probability and correctness of PCLRQ, since the algorithms for HNF and SNF computation succeed unconditionally and are correct.

For the correctness of PDLRQ note that

$$A(\pi(\mathcal{V}))\ (x_v)_{v \in \mathcal{V}}^T = (-n, 1, 0, \ldots, 0)$$

Algorithm 4.16: Class group structure and regulator (RQ)

Input: Δ fundamental discriminant

Output: class number h_Δ, class group invariants s_i, and regulator approximation $R \in \mathbb{Q}$ such that $|R - R_\Delta| < 1$; or FAILURE

PCLRRQ(Δ)

1. Let $c \leftarrow 3\log^2|\Delta|$.
2. Enumerate generating set $\mathcal{G} \leftarrow \mathcal{F}_c$ and let $l = \operatorname{card}\mathcal{G}$.
3. Compute relation vectors $(\mathcal{V}, h, R) \leftarrow$ ESSRLRQ($\Delta, \mathcal{G}$).
4. **if** INVCHECK(Δ, h, R) fails **then print** FAILURE and **exit**.
5. Compute essential part of HNF: $H \leftarrow$ ESSHNF($A(\pi(\mathcal{V})), h$).
6. Compute SNF of essential part: $\operatorname{diag}(s_1, \ldots, s_l) \leftarrow \operatorname{SNF}(H)$.
7. **print** h, s_1, …, s_l, and R.

Algorithm 4.17: Discrete Logarithm (RQ)

Input: Δ fundamental discriminant, reduced ideals $\mathfrak{a}$, and $\mathfrak{b}$

Output: Minimal $n \in \mathbb{N}$ and $\alpha \in \mathcal{K}$ such that $\mathfrak{a} = \alpha\mathfrak{b}^n$; $\emptyset$ if $\mathfrak{a} \notin \langle\overline{\mathfrak{b}}\rangle$; or FAILURE

PDLRQ(Δ)

1. Let $c \leftarrow 3\log^2|\Delta|$.
2. Enumerate generating set $\mathcal{G} \leftarrow \{\mathfrak{b}, \mathfrak{a}\} \cup \mathcal{F}_c$ and let $l = \operatorname{card}\mathcal{G}$.
3. Compute relation vectors $(\mathcal{V}, h, R) \leftarrow$ ESSRLRQ($\Delta, \mathcal{G}$).
4. **if** INVCHECK(Δ, h, R) fails **then print** FAILURE and **exit**.
5. Compute essential part of HNF: $H = (h_{i,j}) \leftarrow$ ESSHNF($A(\pi(\mathcal{V})), h$).
6. **if** $h_{2,2} = 1$ **then** $n \leftarrow h_{1,1} - h_{1,2}$,
7. **else print** $\emptyset$ and **exit**.
8. Compute ε_Δ in compact representation from R.
9. Let $\mathbf{v} \leftarrow (-n, 1, 0, \ldots, 0)$, and compute $\mathbf{x} = (x_v)_{v\in\mathcal{V}} \leftarrow$ DSOLV($A(\pi(\mathcal{V})), \mathbf{v}$).
10. Let $\alpha \leftarrow \prod_{v\in\mathcal{V}} \alpha_v^{x_v}$ and find $t \in \mathbb{Z}$ such that $-1 < \operatorname{Log}(\alpha\varepsilon_\Delta^t) < R + 1$.
11. Let $(\mathfrak{c}, \gamma) \leftarrow$ RHO0($\mathfrak{a}\mathfrak{b}^{-n}$).
12. Compute the compact representation of $\eta \leftarrow \alpha\gamma\varepsilon_\Delta^t\gamma^{-1}$.
13. **print** n, and $\eta\gamma^{-1}$.

implies that

$$(\alpha) = \left(\prod_{v \in \mathcal{V}} \alpha_v^{x_v} \right) = \mathfrak{a}\mathfrak{b}^{-n}.$$

The ideal $\mathfrak{c}$ is reduced, and has henceforth small norm so that the compact representation of a generator can be computed in polynomial time.

We turn to the run-time analysis.

The generating set $\mathcal{G}$ can be obtained in polynomial time.

By Proposition 4.45, the computation of the relation vectors takes time $L_{|\Delta|}(\frac{1}{2}, 3z + o(1))$ and produces a set $\mathcal{V}$ of relations with no more than $L_{|\Delta|}(\frac{1}{2}, z + o(1))$ elements. It succeeds in producing a generating set $\mathcal{V}$ for an essentially full relation lattice with probability larger than $\Omega(1/\log^2 \Delta)$.

InvCheck takes polynomial time.

The computation of the essential part of the HNF of $A(\pi(\mathcal{V}))$ is computed in time $O^{\sim}(nm^2) = L_{|\Delta|}(\frac{1}{2}, 3z+o(1))$ where n is the dimension of the integral part of the relation lattice, and $m = \operatorname{card} \mathcal{V}$ is the number of relations computed.

The computation of the invariants of Cl_Δ from the essential part H of the HNF of $A(\pi(\mathcal{V}))$ takes polynomial time, since H is an $k \times k$ matrix and $k = 3\log^2|\Delta|$.

We finish the run-time analysis with the steps 8 through 13 of PDLRQ.

Step 9 takes time $L_\Delta(\frac{1}{2}, 3z+o(1))$. The vector $\mathbf{x}$ can be extracted actually already in step 5.

The time needed for computing t in step 10 is dominated by the time required to compute high-precision approximations to R_Δ and $\operatorname{Log} \alpha$. The precision depends linearly on $\log\|\mathbf{x}\|_\infty$ which is bounded by $O^{\sim}(n)$ due to Proposition 5.5, and on $\log \operatorname{card} \mathcal{V}$ which is $O^{\sim}(1)$. By Lemma 2.23 the time needed for these Log computations is hence $L_\Delta(\frac{1}{2}, z + o(1))$.

The same observation applies to obtaining a compact representation of $\eta = \alpha\varepsilon_\Delta^t\gamma^{-1}$ in step 12. Note that

$$(\alpha\gamma\varepsilon_\Delta^t) = (\alpha\gamma) = \gamma \cdot \mathfrak{a}\mathfrak{b}^{-n} = \mathfrak{c}$$

implies that the norm of $\eta = \alpha\gamma\varepsilon_\Delta^t$ is small. Since we also have $0 \le \operatorname{Log} \eta < R$ we are thus assured that a short compact representation of η exists and can be computed in polynomial time.

We conclude that PDLRQ executes in time $L_\Delta(\frac{1}{2}, 3z + o(1))$. ∎

Kapitel 5

Computing the HNF

In this appendix we will show how to compute the Hermit Normal Form of a large relation matrix. All matrices in this chapter are integral, i.e., have entries in $\mathbb{Z}$. For a good overview over algorithms computing the Hermite and Smith Normal Form of general integral matrices see Storjohann's thesis [Sto00]. For a comparison of the HNF algorithm given here with another one proposed in the context of index calculus computations, see [Vol03].

Definition 5.1 An $m \times n$-matrix $A = (a_{i,j})$ is in Hermite Normal Form if it is upper triangular, all entries outside the principal minor are zero, all other entries are non-negative and we have $a_{i,j} < a_{i,i}$ if $j > i$.

In each equivalence class of $m \times n$-matrices under multiplication by unimodular matrices from the right, there is exactly one matrix in Hermite Normal Form (HNF), see, e.g., [PZ89]. If A is any matrix, then we will denote the HNF matrix in its class by $H(A)$. We will also say that $H(A)$ is the HNF of A.

Relation matrices have HNF of a particularly simple form.

Definition 5.2 $H = (h_{i,j})$ be an $m \times n$ matrix in HNF. Let $l \in \mathbb{N}$ be minimal such that $h_{i,i} = 1$ for all i satisfying $l < i \leq n$. The $l \times l$-submatrix $A = (h_{i,j})_{1 \leq i,j \leq l}$ is called the *essential part* of H.

In this situation we will also say that l is the essential size of H.

The following two lemmata will show how to reduce the computation of the HNF of matrices to the solution of Diophantine systems.

We list two trivial lemmata using the following notation for the $\mathbb{Z}$-linear maps

$$\pi_i : \mathbb{Z}^n \longrightarrow \mathbb{Z}^i : (x_1, \ldots, x_n) \longmapsto (x_i, \ldots, x_n),$$
$$\iota_i : \mathbb{Z} \longrightarrow \mathbb{Z}^i : x \longmapsto (x, 0, \ldots, 0).$$

For any lattice $\mathcal{L} \subset \mathbb{Z}^n$, we write $\mathcal{L}_i = \pi_i(\mathcal{L})$, and choose $h_i = h_i(\mathcal{L}) \in \mathbb{Z}_{>0}$ such that $\iota_i^{-1}(\mathcal{L}_i) = h_i \cdot \mathbb{Z}$.

Lemma 5.3 *For any matrix $A \in \mathbb{Z}^{n\times m}$ with full rank and Hermite basis $H = (h_{ij})$, we have $h_{ii} = h_i(\mathcal{L}(A))$.*

For any matrix $A \in \mathbb{Z}^{n\times m}$ with columns $\mathbf{v}_1, \ldots, \mathbf{v}_m$ we denote the matrix with columns $\pi_i(\mathbf{v}_j)$ by A_i. Denote the last column of the $i \times i$ identity matrix by $\mathbf{e}_i$.

Lemma 5.4 *For any matrix $A \in \mathbb{Z}^{n\times m}$ we have*

$$h_i(\mathcal{L}(A)) = \min\{\, d \mid \exists \mathbf{v} \in \mathbb{Z}^m \text{ such that } A_i \cdot \mathbf{v} = d \cdot \mathbf{e}_i \,\}.$$

The last lemma can be re-phrased by saying that the i-th diagonal entry of the Hermite basis of matrix A is the minimal denominator of a solution of $A_i \cdot \mathbf{v} = \mathbf{e}_i$ over $\mathbb{Q}$.

In consequence of the preceding two lemmata we can obtain the determinant of a matrix A with full rank and Hermite basis H with essential part of size l by multiplying the minimal denominators of l systems of linear equations, $\det \mathcal{L}(A) = \det H = \prod_{i=n-l+1}^{n} h_i(\mathcal{L}(A))$.

Mulders and Storjohann have proposed an algorithm for the solution of such a Diophantine system.

Proposition 5.5 *There is an algorithm, called* DSolv *with the following properties:*

On input of any $m \times n$ matrix A, a vector $\mathbf{b} \in \mathbb{Z}^n$ and $\epsilon > 0$ it succeeds with probability exceeding $1 - \epsilon$ and does the following:

- *If the system $A\mathbf{x} = \mathbf{b}$ admits a solution over $\mathbb{Q}$, and d is the minimal denominator of such a solution, then it prints d and a vector $\mathbf{x} \in \mathbb{Z}-$ such that $A\mathbf{x} = d \cdot \mathbf{b}$.*
- *If the system $A\mathbf{x} = \mathbf{b}$ does not admit a solution over $\mathbb{Q}$ it prints $\emptyset$.*

In case it does not succeed, DSolv *prints* Failure.

DSolv *executes in time bounded by $O^{\sim}(nmr \log(1/\epsilon))$, where r is the rank of A.*

The solution vector $\mathbf{x}$ found in case of success satisfies $\log\|\mathbf{x}\|_\infty = O^{\sim}(n \log \beta)$ where $\beta > \|A\|_\infty, \|\mathbf{b}\|_\infty$.

The lower order terms suppressed by the Soft-Oh notation are logarithmic in n, m, r, and $\log \beta$.

The idea of the algorithm was first presented in [MS99]. A version called SpecialMinimalSolution which also yields a certificate for the correctness of its results can be found in [MS00].

From the preceding discussion we obtain an algorithm for the computation of the determinant of the HNF of a matrix for which a bound for its essential size is known. It is shown on the facing page.

Starting from the determinant it is easy to compute the full HNF by using, e.g., Hafner and McCurley's HNF algorithm, see [HM91].

Algorithm 5.1: Det. of matrix with known size of essential part

Input: Matrix $A \in \mathbb{Z}^{n\times m}$,
bound l on the size of the essential part of A,
allowed error probability ϵ
Output: Determinant h of A

EssDet(A, l)
1. **for** $i = 1$ **to** l
2. Compute minimal denominator h_i of $A_i \cdot \mathbf{v} = \mathbf{e}_i$ calling DSolv with allowed error $-\log_2(1-\epsilon)/(2l)$.
3. **if** any of the calls to DSolv fails **then print** Failure and **exit**.
4. **print** determinant of $\mathcal{L}(A)$: $h = \prod_{i=1}^{l} h_i$.

It suffices, however, to compute just the essential part of the HNF. This takes little more effort than computing just the determinant.

Algorithm 5.2: HNF of matrix with known size of essential part

Input: Matrix $A \in \mathbb{Z}^{n\times m}$,
bound l on the size of the essential part of A
Output: Essential part of Hermite normal form H of A

EssHNF(A, l)
1. **for** $i = 1$ **to** l
2. Compute minimal denominator h_i of and a solution $\mathbf{v}_i \in \mathbb{Z}^m$ $A_i \cdot \mathbf{v} = \mathbf{e}_i$ calling DSolv with allowed error $-\log_2(1-\epsilon)/(2l)$.
3. **if** any of the calls to DSolv fails **then print** Failure and **exit**.
4. Let $H = (h_{i,j})$ contain the first l rows of $A * U$ where U is the $m \times l$ matrix containing the $\mathbf{v}_i$ with $i = 1, \ldots, l$, as columns.
5. **for** $i = 2$ **to** l
6. Reduce $h_{i,j} \leftarrow h_{i,j} \bmod h_{i,i}$ for all $j > i$.
7. **print** H.

We state the proposition giving the running time bound for EssDet and EssHNF. In order to simplify the statement we we will let Soft-Oh notation suppress terms logarithmic in the main parameters m, n, l, and $\log\|A\|$.

Proposition 5.6 *Given as input a full rank matrix $A \in \mathbb{Z}^{n\times m}$, some $l \leq n$, and error bound $\epsilon > 0$ the algorithms* EssDet *and* EssHNF *succeed with*

probability larger than $1-\epsilon$*, and do the following:*

If l *is a bound for the essential size of the HNF of* A*, then Algorithm* EssDet *computes the determinant of* A*.*

Algorithm EssHNF *computes the* $l \times l$ *minor of the Hermite normal form* H *of* A*, which contains its essential part as principal minor if its essential size is not larger than* l*.*

In the remaining cases the algorithms pronounce to have failed.

The expected running time of both algorithms is bounded by $O^{\sim}\big(l\, n^2 m\, (\log\|A\|)^2(-\log(1-\epsilon))\big)$*.*

Proof We begin by proving the success probability of both algorithms. It is determined by the success probability of DSolv given in Proposition 5.5. EssDet or EssHNF succeed with the probability that l calls to DSolv do. With the allowable error probability as given in step 2 of EssDet of step 2 we have total success probability

$$\left(1+\frac{\log_2(1-\epsilon)}{2l}\right)^l = \left(1-\frac{-\log(1-\epsilon)}{2l}\right)^{\frac{2l}{-\log_2(1-\epsilon)}\cdot\frac{-\log_2(1-\epsilon)}{2}}$$
$$> \left(\frac{1}{4}\right)^{\frac{-\log_2(1-\epsilon)}{2}} = 1-\epsilon.$$

Correctness of EssDet follows from lemmata 5.3 and 5.4.

We prove the correctness of EssHNF.

Clearly, the matrix $A * U$ computed in step 4 is upper triangular, and all columns lie in $\mathcal{L}(A)$. Column reduction as in step 6 produces an $n \times l$ matrix in HNF. Call this matrix $G = (g_{i,j})$, its column vectors $\mathbf{g}$. The first l rows of G coincide with the output printed by EssHNF. All other $n-l$ rows are zero.

Let H be the matrix containing the first l columns of the HNF of A. We want to prove that $G = H$. Assume to the contrary that they differ. Let i be the smallest index for which $\mathbf{g}_i \neq \mathbf{h}_i$. By Lemma 5.4, we know that $g_{i,i} = h_{i,i}$. Let j be the largest index for which $g_{i,j} \neq h_{i,j}$. Since also $g_{j,j} = h_{j,j}$, and the matrices are column reduced we have

$$|g_{i,j} - h_{i,j}| < h_{j,j}.$$

Since $\mathbf{g}_i - \mathbf{h}_i$ are both in the column space of A, and all entries with index larger than j are zero we have $h_{j,j} \,|\, (g_{i,j} - h_{i,j})$, too. Since $g_{i,j}$ and $h_{i,j}$ were assumed to differ this is a contradiction.

Thus we have established that EssHNF prints the $l \times l$ principal minor of the HNF of A. Clearly, if the essential size of the HNF of A does not exceed l, then EssHNF prints a matrix of which the essential part is a principal minor.

The run-time bound follows immediately from Proposition 5.5. ∎

The run-time bound can be improved by optimized matrix multiplication routines, as given e.g. in [CW90], and FFT-based integer arithmetic.

Literaturverzeichnis

[Abe94] Christine Abel. *Ein Algorithmus zur Berechnung der Klassenzahl und des Regulators reellquadratischer Ordnungen.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994. German.

[Bac90] Eric Bach. Explicit bounds for primality testing and related problems. *Mathematics of Computation*, 55(191):355–380, 1990.

[BB87] J. M. Borwein and P.B. Borwein. *Pi and AGM.* Wiley, New York, 1987.

[BB99] Ingrid Biehl and Johannes Buchmann. An analysis of the reduction algorithms for binary quadratic forms. In Peter Engel and Halyna M. Syta, editors, *Voronoi's Impact on Modern Science, Kyiv, Ukraine 1998*, pages 71–98. National Academy of Sciences of Ukraine, 1999.

[BD91a] Johannes Buchmann and Stephan Düllmann. On the computation of discrete logarithms in class groups (extended abstract). In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 134–139. Springer-Verlag, 1991.

[BD91b] Johannes Buchmann and Stephan Düllmann. A probabilistic class group and regulator algorithm and its implementation. In Attila Pethö, Michael E. Pohst, Hugh C. Williams, and Horst Günter Zimmer, editors, *Computational Number Theory*, pages 53–72. Walter de Gruyter Publishers, 1991.

[Ber02] Daniel J. Bernstein. How to find small factors of integers. 2002. to appear in Mathematics of Computation, see `http://cr.yp.to/papers/sf.ps`.

[BH96] Johannes Buchmann and Christine S. Hollinger. On smooth ideals in number fields. *Journal of Number Theory*, 59(1):82–87, 1996.

[BJT97] Johannes Buchmann, Michael J. Jacobson, Jr., and Edlyn Teske. On some computational problems in finite abelian groups. *Mathematics of Computation*, 66(220):1663–1687, 1997.

[BP98] Renet Lovorn Bender and Carl Pomerance. Rigorous discrete logarithm computations in finite fields via smooth polynomials. In *Computational perspectives on number theory (Chicago, IL, 1995)*, volume 7 of *AMS/IP Stud. Adv. Math.*, pages 221–232. Amer. Math. Soc., Providence, RI, 1998.

[Bre76] R.P. Brent. Fast multiple-precision evaluation of elementary functions. *J. Assoc. Comput. Mach.*, 23:242–251, 1976.

[BTV04] Johannes Buchmann, Tsuyoshi Takagi, and Ulrich Vollmer. Number field cryptography. volume 41 of *Fields Institute Communications*, pages 111–125. Fields Institute, American Mathematical Society, 2004.

[BTW95] Johannes Buchmann, Christoph Thiel, and Hugh C. Williams. Short representation of quadratic integers. In Wieb Bosma and Alf J. van der Poorten, editors, *Computational Algebra and Number Theory, Sydney 1992*, volume 325 of *Mathematics and its Applications*, pages 159–185. Kluwer Academic Publishers, 1995.

[Buc90] Johannes Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. In Catherine Goldstein, editor, *Séminaire de Théorie des Nombres, Paris 1988–1989*, volume 91 of *Progress in Mathematics*, pages 27–41. Birkhäuser, 1990.

[BW88] Johannes Buchmann and Hugh C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, 1988.

[BW90] Johannes A. Buchmann and Hugh C. Williams. A key exchange system based on real quadratic fields. extended abstract. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 335–343. Springer-Verlag, 1990.

[Cox89] D.A. Cox. *Primes of the form $x^2 + ny^2$*. Wiley, New York, 1989.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9(3):251–280, 1990.

[Ham02] Safuat Hamdy. *Über die Sicherheit und Effizienz kryptografischer Verfahren mit Klassengruppen imaginär-quadratischer Zahlkörper.* PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2002. `http://www.informatik.tu-darmstadt.de/ftp/pub/TI/reports/hamdy.diss.pdf`.

[HM89] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American Mathematical Society*, 2(4):837–850, 1989.

[HM91] J.L. Hafner and K.S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM J. Comput.*, 20:1068–1083, 1991.

[HM00] Safuat Hamdy and Bodo Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. Technical Report TI-4/00, Technische Universität Darmstadt, Fachbereich Informatik, 2000. `http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/`.

[Hua42] Loo-keng Hua. On the least solution of Pell's equation. *Bull. Amer. Math. Soc.*, 48:731–735, 1942.

[Jac99] Michael J. Jacobson, Jr. *Subexponential Class Group Computation in Quadratic Orders*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 1999.

[Jac00] Michael J. Jacobson, Jr. Computing discrete logarithms in quadratic orders. *J. Cryptology*, 13(4):473–492, 2000.

[Kra22] M. Kraitchik. *Theéorie des Nombres*, volume 1. Gauthier-Villars, 1922.

[Len82] Hendrik W. Lenstra, Jr. On the calculation of regulators and class numbers of quadratic fields. In J. V. Armitage, editor, *Journees Arithmetiques, Exeter 1980*, volume 56 of *London Mathematical Society Lecture Notes Series*, pages 123–150. Cambridge University Press, 1982.

[LP92] H.W. Lenstra Jr. and C. Pomerance. A rigorous time bound for factoring integers. *J. Amer. Math. Soc.*, 5:483–516, 1992.

[Mau00] Markus Maurer. *Regulator approximation and fundamental unit computation for real quadratic orders*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 2000.

[MS99] Thom Mulders and Arne Storjohann. Diophantine linear system solving. In Sam Dooley, editor, *International Symposium on Symbolic and Algebraic Computation, ISSAC '99*. ACM Press, 1999.

[MS00] Thom Mulders and Arne Storjohann. Certified linear system solving. Technical report, ETH Zürich, 2000.

[Odl00] Andrew Odlyzko. Discrete logarithms: the past and the future. *Designs, Codes and Cryptography. An International Journal*, 19(2-3):129–145, 2000. Towards a quarter-century of public key cryptography.

[Oes79] Joseph Oesterlé. Versions effectives du théorème de chebotarev sous l'hypothèse de riemann généralisée. *Astérisque*, 61:165–167, 1979.

[PZ89] M. Pohst and H. Zassenhaus. *Algorithmic Algebraic Number Theory.* CUP, 1989.

[Sch18] Issai Schur. Einige Bemerkungen zur vorstehenden Arbeit des Herrn G. Pólya: Über die Verteilung der quadratischen Reste und Nichtreste. *Nachrichten der Königlichen Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-physikalische Klasse*, pages 30–36, 1918.

[Sch71] Arnold Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, pages 139–144, 1971.

[Sey87] Martin Seysen. A probablistic factorization algorithm with quadratic forms of negative discriminant. *Mathematics of Computation*, 48:757–780, 1987.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, pages 415–440. Amer. Math. Soc., Providence, R.I., 1971.

[Sha72] Daniel A. Shanks. The infrastructure of a real quadratic field and its applications. In *Proceedings of Number Theory Conference, Boulder 1972*, pages 217–224, 1972.

[Sto98] Arne Storjohann. Computing Hermite and Smith normal forms of triangular integer matrices. *Linear Algebra Appl.*, 282(1–3):25–45, 1998.

[Sto00] Arne Storjohann. *Algorithms for Matrix Canonical Forms.* PhD thesis, Swiss Federal Institute of Technology – ETH, 2000.

[Ter00] David C. Terr. A modification of Shanks' baby-step giant-step algorithm. *Mathematics of Computation*, 69(230):767–773, 2000.

[Vol00] Ulrich Vollmer. Asymptotically fast discrete logarithms in quadratic number fields. In Wieb Bosma, editor, *Algorithmic Number Theory, ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 581–594. Springer-Verlag, 2000.

[Vol02] Ulrich Vollmer. An accelerated Buchmann algorithm for regulator computation in real quadratic fields. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, ANTS-V*, volume 2369 of *Lecture Notes in Computer Science*, pages 148–162. Springer-Verlag, 2002.

[Vol03] Ulrich Vollmer. A note on the Hermite basis computation of large integer matrices. In J. Rafael Sendra, editor, *International Symposium on Symbolic and Algebraic Computation, ISSAC '03*, pages 255–257. ACM Press, 2003.

[Wil85] Hugh C. Williams. Continued fractions and number-theoretic computations. *Rocky Mountain J. Math.*, 15(2):621–655, 1985. Number theory (Winnipeg, Man., 1983).

[WM68] A. E. Western and J. C. P. Miller. *Tables of indices and primitive roots*. Royal Society Mathematical Tables, Vol. 9. Published for the Royal Society at the Cambridge University Press, London, 1968.

Curriculum Vitae

Education

1979-83	High School Diploma, **Specialized Schol for Mathematics "Heinrich Hertz"**, Berlin
1985-90	University Diploma in Mathematics (M.A. equivalent), **Friedrich-Schiller-Universität**, Jena
1991-94	PhD program in mathematics, **Massachussetts Institute of Technology**, Cambridge, MA, USA
1999-2003	PhD in Computer Science, **Technische Universität Darmstadt**

Work experience

1983-85	Operator and Programmer, Computing Center, **Funkwerk Köpenick**, Berlin
1990-91	Research Assistant, Department of Mathematics, **Friedrich-Schiller-Universtität**, Jena
1992-94	Teaching Assistant, Department of Mathematics, **Massachussetts Institute of Technology**
1994-95	Lecturer, Department of Computer Engineering, **Escuela Politécnica Nacional**, Quito, Ecuador
1995-99	Research and Teaching Assistant, Department of Psychology and Education, **Ludwig-Maximilians-Universität**, München
1999-	Research Assistant, Department of Computer Science, **Technische Universität Darmstadt**