MEMOIRS OF THE EUROPEAN MATHEMATICAL SOCIETY

Alexander Bors
Daniel Panario
Qiang Wang

Functional Graphs of Generalized Cyclotomic Mappings of Finite Fields

MEMS Vol. 23 / 2026

EMS PRESS


EMS
PRESS

Memoirs of the European Mathematical Society

Edited by

The *Memoirs of the European Mathematical Society* publish outstanding research contributions in individual volumes, in all areas of mathematics and with a particular focus on works that are longer and more comprehensive than usual research articles.

The collection's editorial board consists of the editors-in-chief of the *Journal of the European Mathematical Society* and the *EMS Surveys in Mathematical Sciences*, along with editors of book series of the publishing house of the EMS as well as other distinguished mathematicians.

All submitted works go through a highly selective peer-review process.

Latest titles in this series:

G. Cotti, *Cyclic Stratum of Frobenius Manifolds, Borel–Laplace (α, β)-Multitransforms, and Integral Representations of Solutions of Quantum Differential Equations*
A. Kostenko, N. Nicolussi, *Laplacians on Infinite Graphs*
A. Carey, F. Gesztesy, G. Levitina, R. Nichols, F. Sukochev, D. Zanin, *The Limiting Absorption Principle for Massless Dirac Operators, Properties of Spectral Shift Functions, and an Application to the Witten Index of Non-Fredholm Operators*
J. Kigami, *Conductive Homogeneity of Compact Metric Spaces and Construction of p-Energy*
A. Buium, L. E. Miller, *Purely Arithmetic PDEs Over a p-Adic Field: δ-Characters and δ-Modular Forms*
M. Duerinckx, A. Gloria, *On Einstein's Effective Viscosity Formula*
R. Willett, G. Yu, *The Universal Coefficient Theorem for C*-Algebras with Finite Complexity*
B. Janssens, K.-H. Neeb, *Positive Energy Representations of Gauge Groups I. Localization*
S. Dipierro, G. Giacomin, E. Valdinoci, *The Lévy Flight Foraging Hypothesis in Bounded Regions*
A. Naor, *Extension, Separation and Isomorphic Reverse Isoperimetry*
N. Lerner, *Integrating the Wigner Distribution on Subsets of the Phase Space, a Survey*
B. Adcock, S. Brugiapaglia, N. Dexter, S. Moraga, *On Efficient Algorithms for Computing Near-Best Polynomial Approximations to High-Dimensional, Hilbert-Valued Functions from Limited Samples*
J. J. Carmona, K. Fedorovskiy, *Carathéodory Sets in the Plane*
D. Abramovich, Q. Chen, M. Gross, B. Siebert, *Punctured Logarithmic Maps*
T. Oh, M. Okamoto, L. Tolomeo, *Stochastic Quantization of the Φ^3_3-Model*
H. Sasahira, M. Stoffregen, *Seiberg–Witten Floer Spectra for $b_1 > 0$*
J. Kaad, D. Kyed, *The Quantum Metric Structure of Quantum* SU(2)
B. Zavyalov, *Almost Coherent Modules and Almost Coherent Sheaves*
J-P. Labesse, B. Lemaire, *La formule des traces tordue pour un corps global de caractéristique $p > 0$*
S. Ma, *Vector-Valued Orthogonal Modular Forms*
Y. Almog, B. Helffer, *On the Stability of Symmetric Flows in a Two-Dimensional Channel*

Alexander Bors
Daniel Panario
Qiang Wang

Functional Graphs of Generalized Cyclotomic Mappings of Finite Fields

EMS PRESS

Authors

Alexander Bors
Vienna, Austria

Email: alexander.bors@gmx.at

Daniel Panario
Carleton University
1125 Colonel By Dr
Ottawa ON K1S 5B6, Canada

Email: daniel@math.carleton.ca

Qiang Wang
Carleton University
1125 Colonel By Dr
Ottawa ON K1S 5B6, Canada

Email: wang@math.carleton.ca

Each volume of the *Memoirs of the European Mathematical Society* is available individually or as part of an annual subscription. It may be ordered from your bookseller, subscription agency, or directly from the publisher via subscriptions@ems.press.

ISSN 2747-9080, eISSN 2747-9099
ISBN 978-3-98547-101-0, eISBN 978-3-98547-601-5, DOI 10.4171/MEMS/23

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Published by EMS Press, an imprint of the

European Mathematical Society – EMS – Publishing House GmbH
Institut für Mathematik
Technische Universität Berlin
Straße des 17. Juni 136
10623 Berlin, Germany

Email: info@ems.press

https://ems.press

Typesetting: AfricanType, Cairo, Egypt
Printed in Germany
♾ Printed on acid free paper

Abstract

The functional graph of a function $g : X \to X$ is the directed graph with vertex set X the edges of which are of the form $x \to g(x)$ for $x \in X$. Functional graphs are studied because they allow one to understand the behavior of g under iteration (i.e., to understand the discrete dynamical system (X, g)), which has various applications, especially when X is a finite field $\mathbb{F}_q$. This memoir is an extensive study of the functional graphs of so-called index d generalized cyclotomic mappings of $\mathbb{F}_q$, which are a natural and manageable generalization of monomial functions. We provide both theoretical results on the structure of their functional graphs and Las Vegas algorithms for solving fundamental problems, such as parametrizing the connected components of the functional graph by representative vertices, or describing the structure of a connected component given by a representative vertex. The complexity of these algorithms is analyzed in detail, and we make the point that for fixed index d and most prime powers q (in the sense of asymptotic density), suitable implementations of these algorithms have an expected runtime that is polynomial in $\log q$ on quantum computers, whereas their expected runtime is subexponential in $\log q$ on a classical computer. We also discuss four special cases in which one can devise Las Vegas algorithms with this kind of complexity behavior over most finite fields that solve the graph isomorphism problem for functional graphs of generalized cyclotomic mappings.

Mathematics Subject Classification (2020). Primary 11T22; Secondary 05C05, 05C25, 05C60, 11A07, 37P25

Keywords. cyclotomic mapping, finite dynamical system, finite field, functional graph, generalized cyclotomic mapping

Acknowledgments. The authors would like to thank Ofir Gorodetsky and the anonymous MathOverflow user "Dr. Pi", who both provided helpful answers to a question posted by the first author on MathOverflow. Furthermore, the authors wish to thank the three anonymous reviewers for their careful reading of this long manuscript and their many helpful comments. In particular, one of them provided a long detailed report with many insightful suggestions that greatly improved this manuscript.

Funding. Alexander Bors was supported by the Natural Sciences and Engineering Research Council of Canada, projects RGPIN-2017-06410 and RGPIN-2018-05328. Daniel Panario was supported by the Natural Sciences and Engineering Research Council of Canada, project RGPIN-2018-05328. Qiang Wang was supported by the Natural Sciences and Engineering Research Council of Canada, project RGPIN-2017-06410.

Contents

Chapter 1

Introduction

A *discrete dynamical system* is a pair (X, g), where X is a set and g is a function $X \to X$. The motivation behind this definition is to think of a complicated system that evolves in discrete time steps (such as a neural network), with X being the set of all states which the system can assume, and $g(x)$ being the successor state of $x \in X$. For this reason, one calls X the *state space* and g the *(state) transition function* of (X, g). When studying a discrete dynamical system (X, g), one is naturally interested in the behavior of g under iteration (i.e., in the function iterates g^n for $n \in \mathbb{N}_0 = \{n \in \mathbb{Z} : n \geq 0\}$). See the monograph [48] for a general introduction to discrete dynamical systems, and [48, Chapter 7] in particular for some examples of practical applications of them.

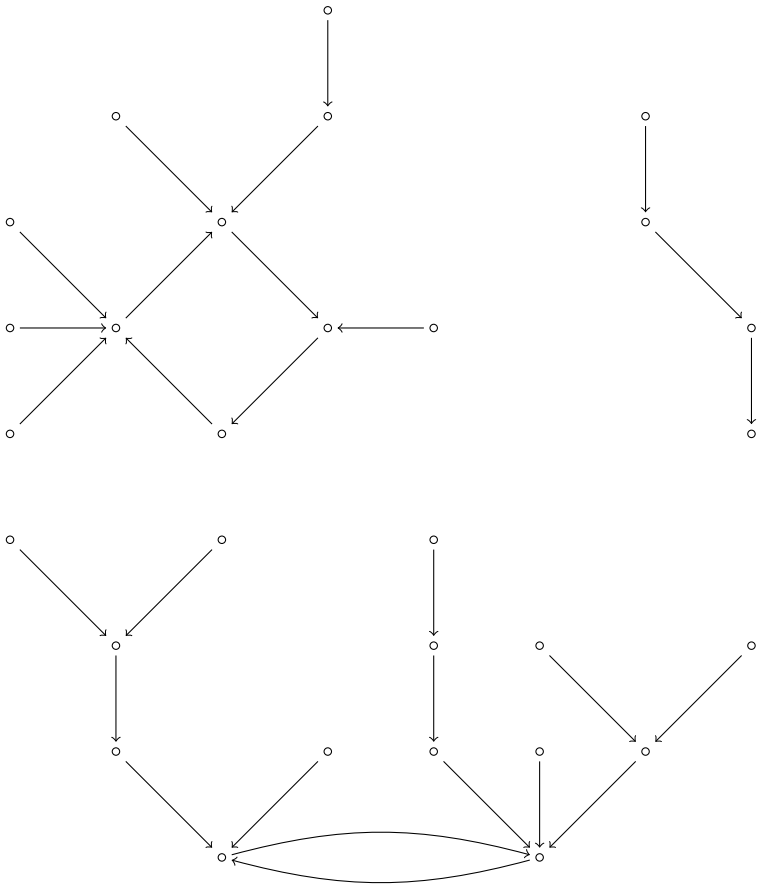
When X is finite, one also calls (X, g) a *finite dynamical system*. Some important special cases with regard to applications are when $X = \mathbb{Z}/m\mathbb{Z}$ and g is a polynomial modulo m (which is used in Pollard's rho algorithm [59]), or when $X = \mathbb{F}_q^n$ (Cartesian power of the finite field $\mathbb{F}_q$), with a particular focus on $q = 2$ in the literature (see [36, 43, 44, 54, 74]). It should be noted that one may identify $\mathbb{F}_q^n$ with $\mathbb{F}_{q^n}$ by fixing an $\mathbb{F}_q$-basis in the latter, so there is in fact no loss of generality when assuming $n = 1$ (i.e., when only considering finite fields themselves as state spaces).

A simple yet remarkable fact when X is finite is that all points $x \in X$ are *pre-periodic under* g, i.e., there exist unique smallest integers $\mathrm{pperl}_g(x) \geq 0$ and $\mathrm{perl}_g(x) \geq 1$, called the *pre-period (length)* and *period (length)* of x under g, respectively, such that $g^{\mathrm{pperl}_g(x)+\mathrm{perl}_g(x)}(x) = g^{\mathrm{pperl}_g(x)}(x)$. A different terminology frequently used in the literature for pre-periodic points is eventually (or ultimately) periodic. In the case $\mathrm{pperl}_g(x) = 0$, one says that x is *periodic under* g (or *g-periodic*); periodic points are also known as purely periodic. The subset of X consisting of all g-periodic points is denoted by $\mathrm{per}(g)$. A point in X that is not g-periodic is called *transient under* g (or g-transient). Various stochastic parameters of random functions $X \to X$ that are of interest for the study of finite dynamical systems, such as the expected pre-period and period length of a point, were determined in [23].

An important means of visualizing a discrete dynamical system (X, g), especially when X is finite, is the so-called *functional graph of* g, denoted by Γ_g. This is the directed graph with vertex set X that has an arc (directed edge) $x \to g(x)$ for each $x \in X$, and no other arcs. It is straightforward to show that a directed graph Γ with vertex set X is a functional graph (i.e., is of the form Γ_g for some $g : X \to X$) if and only if each $x \in X$ has out-degree 1 in Γ.

Particularly for *finite* functional graphs Γ_g, one can give the following precise characterization of their shape: a *connected component of* Γ_g is the induced subgraph

of Γ_g on a subset of X that is the vertex set of a connected component of the underlying undirected graph of Γ_g. Each such connected component contains a single cycle of periodic points of g. Apart from those periodic points, the connected component consists precisely of those points which eventually map to the cycle after sufficiently many iterations of g – the *iterated pre-images (under g)* of points on the cycle. For each x on the cycle, the iterated pre-images y of x such that $x = y^{\mathrm{pperl}_g(y)}$ form a directed rooted tree, with root x, that has all of its arcs oriented toward the root. Henceforth, for simplicity, whenever we say "(directed) rooted tree", it means "directed rooted tree in which all arcs are oriented toward the root". Here is a picture to illustrate the situation:



Conversely, each finite digraph of the shape described above is a functional graph, as it is readily verified that all vertices in it have out-degree 1. The study of finite dynamical systems may be understood as the study of finite functional graphs. In this context, it is also noteworthy that in case ψ_1 and ψ_2 are permutations of a finite set, we have $\Gamma_{\psi_1} \cong \Gamma_{\psi_2}$ if and only if ψ_1 and ψ_2 are of the same *cycle type*, i.e.,

they have the same number of cycles of each given length. Formally, the cycle type of a permutation ψ of X, denoted by $\mathrm{CT}(\psi)$, is defined as the unique monomial in $\mathbb{Q}[x_n : n \in \mathbb{N}^+]$, where $\mathbb{N}^+ = \{n \in \mathbb{Z} : n \geq 1\}$, in which the degree of each variable x_n is the number of ψ-cycles of length equal to n. For example, if $X = \{1, 2, \ldots, 9\}$ and $\psi = (1, 2, 3)(4, 5)(6, 7)(8)(9)$, then

$$\mathrm{CT}(\psi) = x_1^2 x_2^2 x_3.$$

Cycle types (and the related notion of cycle indices) are well studied in combinatorics, and studying isomorphism types of functional graphs may be seen as a natural generalization of this to arbitrary functions on finite sets.

Functional graphs of certain classes of functions on finite fields received considerable attention recently, see the papers [33, 50, 57, 61, 62, 71–73] and references therein. A survey of this topic can be found in [49]. Additionally, the papers [18, 58] do not deal explicitly with functional graphs, but with the iteration of functions on finite fields, and their results could be reformulated in terms of functional graphs. In this memoir, we contribute to this line of research by investigating functional graphs of so-called *generalized cyclotomic mappings* in the following sense.

Definition 1.1. Let q be a prime power, and let $d \mid q - 1$. A *generalized cyclotomic mapping of $\mathbb{F}_q$ of index d* is a function $f : \mathbb{F}_q \to \mathbb{F}_q$ with $f(0) = 0$ such that the restriction of f to each coset C_i of the unique index d subgroup of $\mathbb{F}_q^*$ agrees with a monomial function

$$x \mapsto a_i x^{r_i}.$$

More specifically, let ω be a primitive element of $\mathbb{F}_q$ (i.e., a generator of the cyclic multiplicative group $\mathbb{F}_q^*$), and let C be the index d subgroup of $\mathbb{F}_q^*$. The d cosets of C in $\mathbb{F}_q^*$ are of the form

$$C_i = \omega^i C$$

for $i = 0, 1, \ldots, d - 1$. The general form of an index d generalized cyclotomic mapping f of $\mathbb{F}_q$ is

$$f(x) = \begin{cases} 0, & \text{if } x = 0, \\ a_0 x^{r_0}, & \text{if } x \in C = C_0, \\ a_1 x^{r_1}, & \text{if } x \in C_1, \\ \vdots & \vdots \\ a_{d-1} x^{r_{d-1}}, & \text{if } x \in C_{d-1}, \end{cases} \tag{1.1}$$

where $a_i \in \mathbb{F}_q$ and $r_i \in \{0, 1, \ldots, q - 2\}$ for $i = 0, 1, \ldots, d - 1$. These functions are interesting because they generalize monomial mappings (which constitute the special

case $d = 1$) while still being relatively well controlled. From an abstract algebraic point of view, it is noteworthy that monomial functions

$$\mathbb{F}_q^* \to \mathbb{F}_q^*, \quad x \mapsto ax^r,$$

where $a \neq 0_{\mathbb{F}_q}$ necessarily, are *affine maps* of the multiplicative group $\mathbb{F}_q^*$, in the sense that they are compositions of a group endomorphism of $\mathbb{F}_q^*$ (viz., the power function $x \mapsto x^r$) with a (multiplicative) translation

$$x \mapsto ax$$

by a fixed group element a (see also Definition 2.1.15). Hence, generalized cyclotomic mappings in which all coefficients a_i from (1.1) are non-zero may be viewed as "coset-wise affine" functions, and we explore the idea of generalizing the methods and results from this memoir to other (possibly non-abelian) groups in Section 6.4. In this context, we also note that the celebrated Collatz function $g : \mathbb{Z} \to \mathbb{Z}$, given by the formula

$$g(x) = \begin{cases} x/2, & \text{if } x \in 2\mathbb{Z}, \\ 3x + 1, & \text{if } x \in 2\mathbb{Z} + 1, \end{cases}$$

is also a coset-wise affine function, of its respective domain of definition group $\mathbb{Z}$. The reason why we are able to develop a theory for understanding the behavior of generalized cyclotomic mappings under iteration in this memoir (while the analogous task for the Collatz function is wide open) is because generalized cyclotomic mappings preserve the associated partition of $\mathbb{F}_q$ into the cosets C_i and the singleton set $\{0_{\mathbb{F}_q}\}$ (and, relatedly, they form a semigroup under function composition) – see also the distinction between the two concepts introduced in Definition 6.4.1 (2,3).

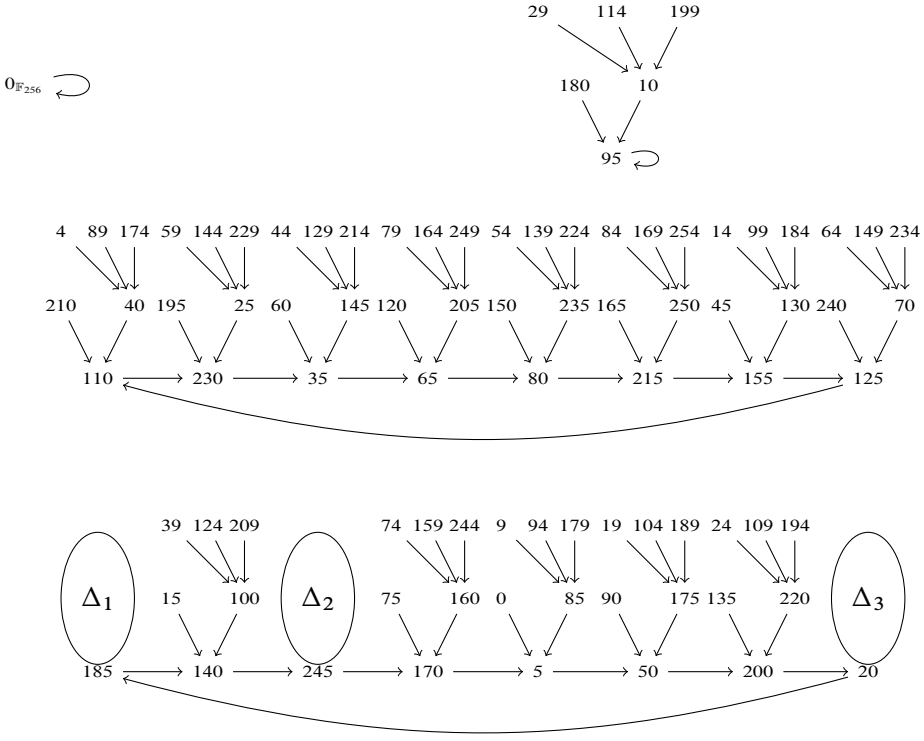
We observe that a given generalized cyclotomic mapping of $\mathbb{F}_q$ may have several possible indices, and that *every* function $f : \mathbb{F}_q \to \mathbb{F}_q$ with $f(0) = 0$ is a generalized cyclotomic mapping of $\mathbb{F}_q$ of index $q - 1$, though the study of generalized cyclotomic mappings is mostly focused on small values of d. For $d = q - 1$, known methods of handling generalized cyclotomic mappings, such as [15, Algorithm 1], are essentially the trivial brute-force approaches. Apart from [15], generalized cyclotomic mappings were also studied in [80, 81, 87]. An important special case is when $r_i = r$ for all i; then one speaks of an *r-th order cyclotomic mapping of $\mathbb{F}_q$ of index d*, and those functions were studied, e.g., in [24, 56, 78, 79, 82].

Our goal in this memoir is to develop algorithms that answer fundamental questions concerning the structure of the functional graph Γ_f of a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$, specified in the form (1.1). For example, let ω be any fixed primitive element of $\mathbb{F}_{256}$, and consider the following index 5 generalized

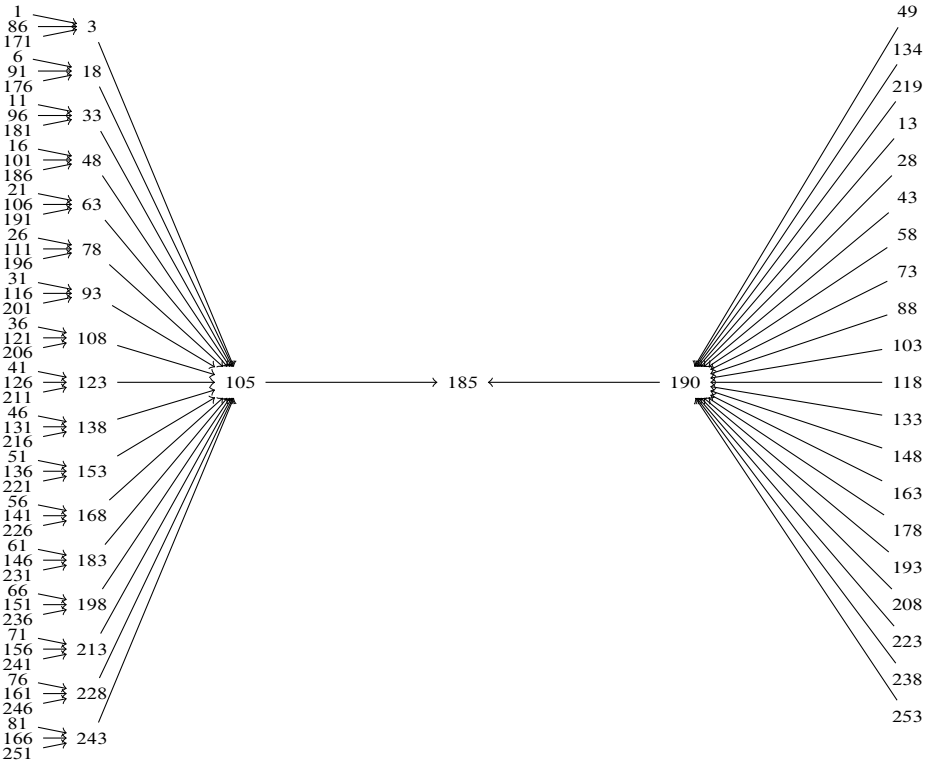
cyclotomic mapping f of $\mathbb{F}_{256}$.

$$f(x) = \begin{cases} 0, & \text{if } x = 0, \\ \omega^5 x^9, & \text{if } x \in C_0, \\ x^3, & \text{if } x \in C_1, \\ x^{17}, & \text{if } x \in C_2, \\ \omega^3 x^{34}, & \text{if } x \in C_3, \\ \omega^4 x^9, & \text{if } x \in C_4. \end{cases} \tag{1.2}$$

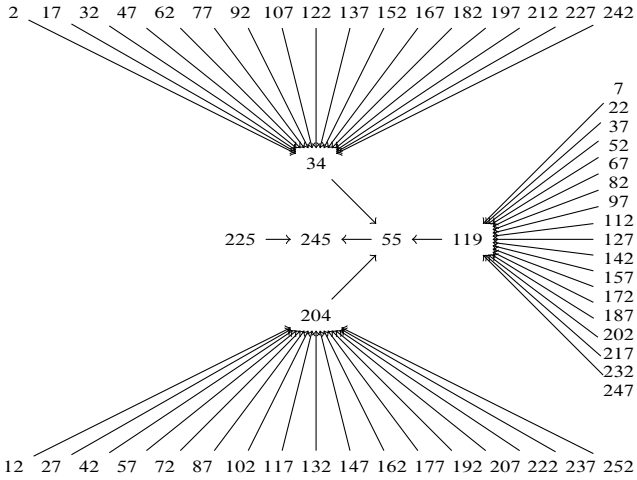
The functional graph Γ_f has 256 vertices, and one can understand its structure by drawing it, which we do below. In this drawing, a vertex labeled $n \in \{0, 1, \ldots, 254\}$ corresponds to the field element $\omega^n \in \mathbb{F}_q^*$ (in particular, the label 0 corresponds to the field element $\omega^0 = 1_{\mathbb{F}_{256}}$), whereas the vertex representing the field element 0 is labeled by $0_{\mathbb{F}_{256}}$. It turns out that Γ_f has four connected components, and in one of them (the fourth one in our order of drawing), the rooted trees glued to three particular f-periodic points on the unique cycle in that connected component are relatively large and thus drawn separately; we mark those rooted trees with Δ_k for $k \in \{1, 2, 3\}$ in the schematic drawing of the corresponding connected component.



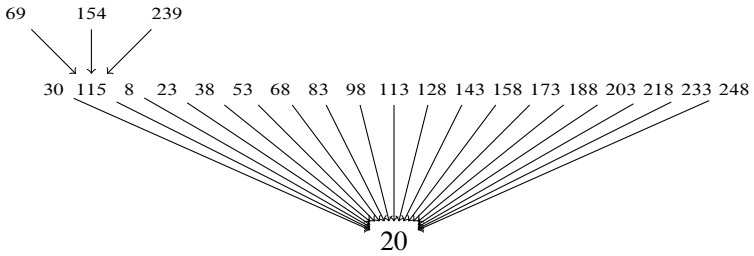
The rooted tree Δ_1 looks as follows.



The rooted tree Δ_2 looks as follows.



The rooted tree Δ_3 looks as follows.



Of course, this approach of understanding Γ_f by drawing it becomes intractable for large values of q, as its complexity is at least linear in q (i.e., exponential in $\log q$). The aim of our algorithms is to obtain an understanding of the structure of Γ_f without needing to draw it vertex by vertex. A detailed complexity analysis of those algorithms, which we carry out in Chapter 5, shows that for asymptotically almost every finite field and fixed index d, our algorithms have implementations with polynomial complexity (in $\log q$) on quantum computers, and implementations with subexponential complexity on classical computers. In the remainder of this introduction, we discuss the main ideas underlying our algorithms. We also note that we revisit the example (1.2) in Section 4.2, where we derive the structure of its functional graph with our methods.

The first step in understanding the functional graph Γ_g of any function $g : X \to X$ is to obtain a suitable parametrization of the connected components of Γ_g. The following notion is helpful in that regard.

Definition 1.2. Let X be a finite set, and $g : X \to X$. A *cycle representatives and lengths list* (or *CRL-list* for short) *of* g is a (finite) set $\mathcal{L} \subseteq X \times \mathbb{N}^+$ with the following properties:

(1) The first entries of the ordered pairs in $\mathcal{L}$ form a system of representatives for the cycles of g on its periodic points.

(2) If $(r, l) \in \mathcal{L}$, then l is the cycle length of r under g.

Remark 1.3. When g is a function on a finite set, it is easy to determine the cycle type of the restriction $g_{|\mathrm{per}(g)}$ from any CRL-list $\mathcal{L}$ of g. Namely,

$$\mathrm{CT}(g_{|\mathrm{per}(g)}) = \prod_{(r,l)\in\mathcal{L}} x_l.$$

A CRL-list of g can thus be seen as a refinement of $\mathrm{CT}(g_{|\mathrm{per}(g)})$.

We recall from above that each connected component of Γ_g contains precisely one cycle of g on its periodic points. This means that a CRL-list of g also gives a parametrization of the connected components of g via representative vertices, along

with the basic information how long the cycle of each representative is. Let us give some more details on how to obtain $\mathfrak{L}$ when $X = \mathbb{F}_q$ and g is an index d generalized cyclotomic mapping f of $\mathbb{F}_q$.

We already introduced the notation $C_i = \omega^i C$ for $i \in \{0, 1, \ldots, d-1\}$ to denote the cosets of C in $\mathbb{F}_q^*$. Let us additionally set $C_d := \{0_{\mathbb{F}_q}\}$. Then the sets C_i for $i = 0, 1, \ldots, d$ form a partition of $\mathbb{F}_q$ that is preserved by f in the sense that f maps blocks of this partition to other such blocks (not necessarily surjectively). In other words, there is a unique function $\bar{f} : \{0, 1, \ldots, d\} \to \{0, 1, \ldots, d\}$, which we call *induced by* f, such that $f(C_i) \subseteq C_{\bar{f}(i)}$ for each $i = 0, 1, \ldots, d$. We note in particular that $\bar{f}(d) = d$, and that $\bar{f}^{-1}(\{d\}) = \{d\}$ unless at least one of the coefficients a_i in (1.1) is 0.

Setting $s := (q-1)/d = |C|$, we may view each coset $C_i = \omega^i C$ for $i = 0, 1, \ldots, d-1$ as a copy of the cyclic group $\mathbb{Z}/s\mathbb{Z}$ (with underlying set $\{0, 1, \ldots, s-1\}$ and modular addition as its group operation) via the bijection $\iota_i : \mathbb{Z}/s\mathbb{Z} \to C_i$, $x \mapsto \omega^{i+dx}$. As such, f may be viewed as a function that maps between copies of $\mathbb{Z}/s\mathbb{Z}$ (as well as a unique singleton block). More specifically, if $i \in \{0, 1, \ldots, d-1\}$ and $a_i \neq 0$, and if we write $a_i = \omega^{e_i}$, then we have $d \mid e_i + r_i i - \bar{f}(i)$ necessarily, and f maps $\omega^{i+dx} \in C_i$ to

$$a_i(\omega^{i+dx})^{r_i} = \omega^{e_i + r_i i + r_i dx} = \omega^{\bar{f}(i) + d\cdot(\frac{e_i + r_i i - \bar{f}(i)}{d} + r_i x)} \in C_{\bar{f}(i)}.$$

This means that under the identifications of C_i and $C_{\bar{f}(i)}$ with $\mathbb{Z}/s\mathbb{Z}$, the restriction of f to C_i corresponds to the affine function $A_i : x \mapsto r_i x + (e_i + r_i i - \bar{f}(i))/d$ of $\mathbb{Z}/s\mathbb{Z}$.

In summary, f consists essentially of affine functions mapping between copies of $\mathbb{Z}/s\mathbb{Z}$, though some copies of $\mathbb{Z}/s\mathbb{Z}$ may also be constantly mapped into $C_d = \{0\}$ by f, in case the corresponding coefficient a_i is 0. We note that if all a_i are non-zero, then the way $f_{|\mathbb{F}_q^*}$ preserves the partition of $\mathbb{F}_q^*$ into the cosets C_i for $i \in \{0, 1, \ldots, d-1\}$ is analogous to the way the elements of the imprimitive permutational wreath product $\mathrm{Sym}(C) \wr \mathrm{Sym}(d)$ (where $\mathrm{Sym}(X)$ and $\mathrm{Sym}(n)$ denote the symmetric group on the set X and on $\{0, 1, \ldots, n-1\}$, respectively) preserve this partition. In fact, the definition of an imprimitive permutational wreath product naturally extends to one of an *imprimitive wreath product of transformation semigroups* such that $f_{|\mathbb{F}_q^*}$ is an element of the imprimitive wreath product of C^C (the transformation semigroup of all functions $C \to C$) with $\{0, 1, \ldots, d-1\}^{\{0,1,\ldots,d-1\}}$, and $\bar{f}_{|\{0,1,\ldots,d-1\}}$ is the projection of f to $\{0, 1, \ldots, d-1\}^{\{0,1,\ldots,d-1\}}$. Wreath products of transformation semigroups have been studied before and play a central role in algebraic automata theory, though the notion used in that theory is the natural generalization of *primitive* permutational wreath products [34, pp. 55f.].

In any case, these ideas allow us to easily reduce the determination of a CRL-list $\mathfrak{L}$ of f to the determination of CRL-lists of affine functions on $\mathbb{Z}/s\mathbb{Z}$ – see

Section 3.1 for the details of this. CRL-lists of affine functions of finite cyclic groups are determined in Section 2.3.

The remainder of our algorithmic approach is concerned with understanding, for each given $(r, l) \in \mathcal{L}$, the isomorphism type of the connected component of Γ_f containing r. We recall from above that this connected component is essentially obtained by glueing certain directed rooted trees to the vertices on the cycle. Let us introduce the following precise notation.

Definition 1.4. Let Γ be a finite functional graph with vertex set X, and let g be the unique function $X \to X$ such that $\Gamma = \Gamma_g$. For each $x \in X$, we define $\mathrm{Tree}_\Gamma(x)$, the so-called *tree above x in Γ*, as follows.

(1) If x is g-transient, we define $\mathrm{Tree}_\Gamma(x)$ as the induced subgraph of Γ on the set

$$\{x\} \cup \{y \in \mathrm{V}(\Gamma) : y \neq x, \text{ and } g^k(y) = x \text{ for some } k = k(y) \geq 1\}.$$

(2) If x is g-periodic, we define $\mathrm{Tree}_\Gamma(x)$ as the induced subgraph of Γ on the set

$$\{x\} \cup \bigcup \{\mathrm{V}(\mathrm{Tree}_\Gamma(y)) : y \text{ is } g\text{-transient and } g(y) = x\},$$

with the convention that in case $g(x) = x$, the loop at x is deleted from $\mathrm{Tree}_\Gamma(x)$.

With this definition, $\mathrm{Tree}_{\Gamma_g}(x)$ is defined for all $x \in X = \mathrm{V}(\Gamma_g)$, and for periodic vertices x, those are the trees that need to be glued to the cycles of g in order to obtain the full connected components of Γ_g.

Necklaces are a well-studied concept in combinatorics. Let us consider vertex-labeled, directed graphs that consist of a single, directed cycle (let us call such a graph a *necklace graph*). Intuitively, one may think of the vertices as beads on a necklace (in the common sense of the word), and the vertex labels represent colors of those beads. An *isomorphism of vertex-labeled digraphs* is a digraph isomorphism preserving vertex labels, and a *necklace* is an isomorphism class of necklace graphs under isomorphism of vertex-labeled digraphs. If $\vec{\mathfrak{x}} = (\mathfrak{x}_0, \mathfrak{x}_1, \ldots, \mathfrak{x}_{L-1})$ is a length-L sequence with entries from a set $\mathfrak{X}$, then we denote by $[\vec{\mathfrak{x}}] = [\mathfrak{x}_0, \mathfrak{x}_1, \ldots, \mathfrak{x}_{L-1}]$ the orbit of $\vec{\mathfrak{x}}$ under the natural action of the cyclic group $\mathbb{Z}/L\mathbb{Z}$ on $\mathfrak{X}^L$. Hence, $[\vec{\mathfrak{x}}]$ consists of those length-L sequences over $\mathfrak{X}$ that can be obtained from $\vec{\mathfrak{x}}$ through cyclic shifts. We also call $[\vec{\mathfrak{x}}]$ the *cyclic sequence associated with* $\vec{\mathfrak{x}}$. We observe that two necklace graphs are isomorphic if and only if their sequences of vertex labels are cyclically equivalent, whence in combinatorics, a necklace is often simply defined as a cyclic sequence (cyclic equivalence class of strings).

The connected components of a functional graph Γ_g of a function $g : X \to X$, where X is a finite set, may be viewed as necklace graphs. Indeed, we take the unique

directed cycle contained in a given connected component as the underlying digraph of the associated necklace graph. The label of a vertex x on that cycle is defined as the rooted tree isomorphism type of $\mathrm{Tree}_{\Gamma_g}(x)$. For example, if we denote by

- $\mathfrak{I}_0$ the digraph isomorphism type of the trivial rooted tree (consisting of a single vertex without arcs);
- $\mathfrak{I}_1$ the most common rooted tree isomorphism type above a periodic vertex in the functional graph of the exemplary generalized cyclotomic mapping f of $\mathbb{F}_{256}$ defined in (1.2) above (i.e., a rooted tree of height 2, where the root has in-degree 2 and one of the two neighbors of the root has in-degree 0, the other has in-degree 3);
- $\mathfrak{I}_2$ the digraph isomorphism type of Δ_3 in the above example (the seemingly chaotic numbering for the $\mathfrak{I}_j$ is chosen such that it matches with Table 4.4 in Section 4.2);
- $\mathfrak{I}_3$ the digraph isomorphism type of Δ_1;
- $\mathfrak{I}_4$ the digraph isomorphism type of Δ_2;

then the four connected components of the example above may be identified with necklace graphs corresponding to the following cyclic sequences of rooted tree isomorphism types (in order of drawing):

- $[\mathfrak{I}_0]$;
- $[\mathfrak{I}_1]$;
- $[\mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1]$;
- $[\mathfrak{I}_3, \mathfrak{I}_1, \mathfrak{I}_4, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_2]$.

With the above convention of identifying connected components of functional graphs with certain necklace graphs, two digraphs that are connected components of finite functional graphs are isomorphic as digraphs if and only if they are isomorphic as necklace graphs (i.e., they represent the same necklace of rooted tree isomorphism types). This means that in order to understand the connected components of Γ_g, we need to understand the associated cyclic sequences of rooted tree isomorphism types.

We note that if the goal is to understand the (undirected graph) isomorphism type of the underlying undirected graph of a connected component of a functional graph, an analogous approach can be used. One needs to replace necklace graphs by *bracelet graphs* (undirected, vertex-labeled cycle graphs), cyclic sequences by *dihedral sequences* (orbits of the natural action of the dihedral group of order $2L$ on $\mathfrak{X}^L$, where the generating reflection acts by writing the sequence in reverse order), and necklaces by *bracelets* (isomorphism classes of bracelet graphs).

Let us next explain our approach for understanding the digraph isomorphism types of the connected components of Γ_g via necklaces of rooted tree isomorphism types in case $X = \mathbb{F}_q$ and g is an index d generalized cyclotomic mapping f of $\mathbb{F}_q$.

For this, we first need to understand the rooted tree above a given (periodic) point. The basic idea is to construct a certain partition $\mathcal{P}_i$ of each coset C_i that "controls" the isomorphism types of rooted trees above the vertices in each of its blocks. Dealing with entire blocks of vertices at once is crucial to ensure that the complexities of our algorithms are not at least linear in the number of vertices q like many general-purpose algorithms for handling graph isomorphism, including Babai's breakthrough quasi-polynomial algorithm from [9].

In order to sketch how the said partition $\mathcal{P}_i$ of C_i is constructed, we need to introduce some more concepts. For a given positive integer m, we define the notion of an *m-(in)congruence* to be an (in)congruence of the form $\nu(x \equiv \mathfrak{b} \pmod{\mathfrak{a}})$, where $\nu \in \{\emptyset, \neg\}$ is a "logical sign", and $\mathfrak{a}, \mathfrak{b}$ are integers with $\mathfrak{a} \geq 1$ and $\mathfrak{a} \mid m$. We subsume these two notions under the name *m-congruential condition*, or *m-CC* for short. Next, we consider the concept of an *arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$*, see point (2) of the following definition.

Definition 1.5. Let m be a positive integer. We identify the elements of $\mathbb{Z}/m\mathbb{Z}$ with their standard representatives in $\{0, 1, \ldots, m-1\}$.

(1) Let $x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j}$ for $j = 1, 2, \ldots, K$ be m-congruences. There is a unique partition of $\mathbb{Z}/m\mathbb{Z}$, which we denote by

$$\mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K),$$

such that each block of this partition is the solution set modulo m of a system of m-CCs of the form

$$\begin{aligned} &\nu_1(x \equiv \mathfrak{b}_1 \pmod{\mathfrak{a}_1}),\\ &\nu_2(x \equiv \mathfrak{b}_2 \pmod{\mathfrak{a}_2}),\\ &\quad\vdots\\ &\nu_K(x \equiv \mathfrak{b}_K \pmod{\mathfrak{a}_K}), \end{aligned} \tag{1.3}$$

where $\nu_1, \nu_2, \ldots, \nu_K \in \{\emptyset, \neg\}$ are logical signs.

(2) A partition $\mathcal{P}$ of $\mathbb{Z}/m\mathbb{Z}$ is called *arithmetic* if it is of the form

$$\mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$$

for a suitable non-negative integer K and suitable m-congruences

$$x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j}$$

for $j = 1, 2, \ldots, K$. If $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$, then we also say that $\mathcal{P}$ is the arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ *spanned by* the congruences $x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j}$ for $j = 1, 2, \ldots, K$.

(3) When $\mathcal{P}$ is an arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$, then the smallest value of $K \in \mathbb{N}_0$ such that $\mathcal{P}$ is spanned by K suitably chosen m-congruences is called the *(arithmetic) complexity of* $\mathcal{P}$, written $\mathrm{AC}(\mathcal{P})$.

(4) When $\mathcal{P}$ is an arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ and a sequence of spanning congruences $(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j})_{j=1,2,\ldots,K}$ has been fixed for $\mathcal{P}$, then we denote for each $\vec{\nu} = (\nu_1, \nu_2, \ldots, \nu_K) \in \{\emptyset, \neg\}^K$ by $\mathcal{B}(\mathcal{P}, \vec{\nu})$ the unique subset of $\mathbb{Z}/m\mathbb{Z}$ that is the solution set of the system (1.3) (this solution set is a block of $\mathcal{P}$ as long as it is non-empty).

Remark 1.6. There are significantly fewer arithmetic partitions of $\mathbb{Z}/m\mathbb{Z}$ than there are partitions in total. Indeed, the total number of (set) partitions of $\mathbb{Z}/m\mathbb{Z}$ is the Bell number B_m, which satisfies

$$B_m \sim \frac{1}{\sqrt{m}}\left(\frac{m}{W(m)}\right)^{m+\frac{1}{2}} \exp\left(\frac{m}{W(m)} - m - 1\right)$$

as $m \to \infty$, where $W(m) \sim \log m$ is the Lambert W function (see [47, Section 1.14, Problem 9]). In particular, as $m \to \infty$,

$$\log B_m \sim -\frac{1}{2}\log m + \left(m + \frac{1}{2}\right)(\log m - \log W(m)) + \frac{m}{W(m)} - m - 1 \sim m \log m.$$

On the other hand, every arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ is spanned by a selection of congruences of the form $x \equiv \mathfrak{b} \pmod{\mathfrak{a}}$, where $\mathfrak{a}$ ranges over the positive divisors of m, and $\mathfrak{b} \in \{0, 1, \ldots, \mathfrak{a} - 1\}$. Because the total number of such congruences is $\sigma(m)$ (the sum of all positive divisors of m), it follows that the number of arithmetic partitions of $\mathbb{Z}/m\mathbb{Z}$ is at most $2^{\sigma(m)}$, and so its natural logarithm is at most

$$\log 2 \cdot \sigma(m) \leq \log 2(\mathrm{e}^{\gamma} + \varepsilon) m \log\log m,$$

where γ denotes the Euler–Mascheroni constant and the second bound follows from a result of Robin [63].

Let us return to our index d generalized cyclotomic mapping f of $\mathbb{F}_q$. We recall that $s = (q-1)/d$ denotes the order (size) of the index d subgroup C of $\mathbb{F}_q^*$. The aforementioned partitions $\mathcal{P}_i$ of the cosets C_i are constructed as arithmetic partitions of $\mathbb{Z}/s\mathbb{Z}$, with which C_i is to be identified via the bijection ι_i introduced above. They have the property that for vertices $x, y \in C_i$ chosen from a common block $\mathcal{B}(\mathcal{P}_i, \vec{\nu})$ of $\mathcal{P}_i$, one has $\mathrm{Tree}_{\Gamma_f}(x) \cong \mathrm{Tree}_{\Gamma_f}(y)$, and this common isomorphism type is denoted by $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu})$. The constructions of the partitions $\mathcal{P}_i$ and of the associated rooted tree isomorphism types $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu})$, which are carried out in detail in Section 3.3, are based on two crucial tools:

- the elementary result Lemma 2.2.2 from Section 2.2; and

- an explicit understanding, developed in Section 3.2 but also based on earlier theory developed in Section 2.1, of the structures of rooted trees in the induced subgraph Γ_{per} of Γ_f on the union of $\{0_{\mathbb{F}_q}\}$ with all cosets C_i, where i is $\bar{f}$-periodic.

Once the $\mathcal{P}_i$ and $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu})$ have been constructed explicitly, in order to understand the isomorphism type of the connected component of Γ_f with representative periodic vertex r, one needs to understand how the cycle moves through the various blocks of the respective coset partitions. Of course, if the cycle length l of r under f is small, one can just enumerate the points on the cycle by brute force, check in which blocks they lie and spell out the corresponding cyclic sequence of rooted trees; this is what we do at the end of the example in Section 4.2. However, if l is large, then one can obtain a more concise description of the cyclic rooted tree sequence via a certain tuple of arithmetic partitions, the blocks of which represent intersections of the cycle of r with blocks of the involved arithmetic partitions $\mathcal{P}_i$. For details on this, see Section 3.4, which builds on Section 2.4.

Here is an overview of our approach for understanding Γ_f.

(1) Determine the induced function $\bar{f}\colon\{0,1,\ldots,d\}\to\{0,1,\ldots,d\}$, and rewrite f into a collection of affine functions that map between $d+1$ sets C_i, each of the form $\mathbb{Z}/s\mathbb{Z}$ or $\{0\}$.

(2) Compute a CRL-list $\mathcal{L}$ for f as specified in Section 3.1, which is based on the results for affine maps of finite cyclic groups from Section 2.3.

(3) For each $i \in \{0,1,\ldots,d-1\}$, compute the arithmetic partition $\mathcal{P}_i$ and associated rooted tree isomorphism types $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu})$, as well as the isomorphism type of $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, as specified in Section 3.3. This requires the theory developed in Sections 2.1 and 3.2.

(4) For each $(r,l) \in \mathcal{L}$, understand the associated cyclic sequence of rooted tree isomorphism types along the cycle of r under f, either

- by listing elements on the cycle of r by brute force, then looking up in which blocks of the relevant arithmetic partitions they lie, or
- by following the approach from Section 3.4, which relies on Section 2.4.

In Chapter 5, where we give a detailed algorithmic complexity analysis, we treat the procedures described in steps (2–4) each as a separate algorithm to be analyzed. We note that in general, f has too many cycles in order for it to be possible to spell out a CRL-list of f element-wise if the procedure is to be efficient (i.e., subexponential in $\log q$); one can, however, obtain a concise parametrization of a CRL-list of f efficiently. Likewise, the approach described in point (4) can be carried out for each given pair (r,l) individually in an efficient manner for asymptotically almost all finite fields $\mathbb{F}_q$, but it is not clear in general how to obtain a "global" understanding of Γ_f efficiently. In fact, the number of distinct isomorphism types of connected components of Γ_f might be superpolynomial in $\log q$ even for fixed d (cf. Prob-

lem 6.3.3), so one would first need to come up with a compact way of parametrizing those isomorphism types. Still, as we will see in Section 5.3, for some special cases of generalized cyclotomic mappings f of $\mathbb{F}_q$, there are algorithms for describing Γ_f as a whole which are efficient for all or at least for "most" q (in an asymptotic density sense). In particular, in those cases, it can be efficiently decided whether the functional graphs of two given generalized cyclotomic mappings of $\mathbb{F}_q$ are isomorphic.

Chapter 6 concludes the memoir with a list of open problems for further research. For the reader's convenience, an extensive index of the notation and terminology appearing in this memoir is given in Tables A.1 and A.2 in Appendix A.

Chapter 2

Preparations

In this chapter, we prove some auxiliary results that are used when discussing the details of our algorithm in Chapter 3.

2.1 Functional graphs of affine maps of finite groups

In this section, we derive some results on functional graphs of affine maps $A : x \mapsto ax + b$ of finite cyclic groups $\mathbb{Z}/m\mathbb{Z}$. We note that these graphs were studied earlier by Deng [21], and we use several of Deng's results and ideas here, as pointed out where appropriate. However, for the reader's convenience, we aim to keep our exposition self-contained. We also observe (at the end of the section) that these results can in fact be generalized to arbitrary finite groups. First, we consider the following concept.

Definition 2.1.1. Let $(\Gamma_j)_{j\in I}$ be a family of digraphs. Their *tensor product*, written $\bigotimes_{j\in I} \Gamma_j$, is the digraph with vertex set $\prod_{j\in I} \mathrm{V}(\Gamma_j)$ having an arc $(y_j)_{j\in I} \to (z_j)_{j\in I}$ if and only if for each $j \in I$, there is an arc $y_j \to z_j$ in Γ_j.

This concept corresponds to Deng's product graph from [21, formula (1) in Section 2].

Remark 2.1.2. If $(g_j)_{j\in I}$ is a family of functions $g_j : X_j \to X_j$, and if $\bigotimes_{j\in I} g_j$ denotes the function

$$\prod_{j\in I} X_j \to \prod_{j\in I} X_j, \quad (y_j)_{j\in I} \mapsto (g_j(y_j))_{j\in I},$$

then

$$\Gamma_{\bigotimes_{j\in I} g_j} = \bigotimes_{j\in I} \Gamma_{g_j}.$$

Due to Remark 2.1.2, the tensor product of digraphs is a useful tool when studying functional graphs of affine maps (in particular of endomorphisms) of finite cyclic groups $\mathbb{Z}/m\mathbb{Z}$. Indeed, if we factor $m = p_1^{v_1} \cdots p_K^{v_K}$, then for each given affine map $A : x \mapsto ax + b$ of $\mathbb{Z}/m\mathbb{Z}$ and each $j \in \{1, 2, \ldots, K\}$, we may consider the reduction A_j of A modulo $p_j^{v_j}$, which is the affine map $A_j : x \mapsto ax + b$ of $\mathbb{Z}/p_j^{v_j}\mathbb{Z}$. By the Chinese remainder theorem, $\mathbb{Z}/m\mathbb{Z}$ is in a natural isomorphism with $\prod_{j=1}^{K} \mathbb{Z}/p_j^{v_j}\mathbb{Z}$, and under this isomorphism, A corresponds to $\bigotimes_{j=1}^{K} A_j$. Hence we obtain the following, which is [21, Theorem 2].

Lemma 2.1.3. *Let $m = p_1^{v_1} \cdots p_K^{v_K}$ be a positive integer with displayed factorization into pairwise coprime prime powers. Let $A : x \mapsto ax + b$ be an affine map of $\mathbb{Z}/m\mathbb{Z}$, and denote by A_j the reduction of A modulo $p_j^{v_j}$ for $j = 1, 2, \ldots, K$. Then $\Gamma_A \cong \bigotimes_{j=1}^{K} \Gamma_{A_j}$.*

Lemma 2.1.3 allows us to reduce many arguments concerning functional graphs of affine maps of $\mathbb{Z}/m\mathbb{Z}$ to the case where $m = p^v$ is a prime power. We note the following interesting dichotomy (see also [21, Lemma 3]).

Proposition 2.1.4. *Let $m = p^v$ be a prime power, and let $A : x \mapsto ax + b$ be an affine map of $\mathbb{Z}/m\mathbb{Z}$.*

(1) *If $p \mid a$, then A has exactly one periodic point x (a fixed point necessarily), and Γ_A is obtained from $\mathrm{Tree}_{\Gamma_A}(x)$ by adding a loop to the root x.*

(2) *If $p \nmid a$, then A is a permutation of $\mathbb{Z}/m\mathbb{Z}$, whence Γ_A is a disjoint union of directed cycles.*

We would like to use these reduction ideas to prove the following theorem.

Theorem 2.1.5. *Let $A : x \mapsto ax + b$ be an affine map of the finite cyclic group $\mathbb{Z}/m\mathbb{Z}$. Then all trees above periodic vertices in Γ_A are isomorphic to each other. In fact, they are all isomorphic to any tree above a periodic vertex in Γ_{μ_a}, where μ_a is the endomorphism $x \mapsto ax$ of $\mathbb{Z}/m\mathbb{Z}$.*

We note that it was proved by Sha [66, Corollary 3.4] that all trees above periodic vertices in Γ_{μ_a} are isomorphic to each other. The statement of Theorem 2.1.5 itself is implicit in Deng's proof of [21, Theorem 11].

Using Proposition 2.1.4, we can derive the following partial result swiftly, in the proof of which we use the notation

$$v_p^{(v)}(n) := \min\{v, v_p(n)\},$$

where $v_p(n)$ is the *p-adic valuation of n*, i.e., the exponent of p in the prime power factorization of the integer n, defined to be ∞ if $n = 0$.

Lemma 2.1.6. *Theorem 2.1.5 holds when $m = p^v$ is a prime power.*

Proof. By [21, Lemma 4], if $v_p^{(v)}(a-1) \le v_p^{(v)}(b)$, then A has a fixed point, which leads to a digraph isomorphism between Γ_A and Γ_{μ_a} (cf. also our Lemma 2.3.3). The result is thus clear by [66, Corollary 3.4].

On the other hand, if $v_p^{(v)}(b) < v_p^{(v)}(a-1)$, then $a \equiv 1 \pmod{p}$, which implies that $p \nmid a$. Therefore, by Proposition 2.1.4 (2), all rooted trees above periodic vertices in Γ_A are trivial (i.e., they are isomorphic to a single vertex without arcs), and so are those trees in Γ_{μ_a}, as required. ∎

Of course, we could now derive Theorem 2.1.5 in its full strength by observing that the property "all rooted trees above periodic vertices are isomorphic" is preserved under taking tensor products of functional graphs. We can, however, obtain an even more detailed result with some extra work, which we carry out. This requires some concepts and results from the first author's paper [13], in which the structure of the trees above periodic vertices in functional graphs of finite group endomorphisms was characterized (thus extending Sha's result [66, Corollary 3.4]).

Definition 2.1.7. Let $\Gamma = (V, E)$ be a finite digraph, and let $x \in V$.

(1) The *dual digraph* Γ^* is obtained from Γ by inverting each arc; formally, $\Gamma^* = (V, E^{-1})$, where $E^{-1} = \{(y, z) \in V^2 : (z, y) \in E\}$ is the inverse relation of E.

(2) A vertex $y \in V$ such that Γ has an edge $x \to y$ is a *successor* or *child of* x.

(3) For each $k \in \mathbb{N}^+$, the *k-th procreation number of* x, written $\mathrm{proc}_k(x) = \mathrm{proc}_k^{(\Gamma)}(x)$, is the number of children y of x such that there is a length $k-1$ directed path $(w_1, w_2, \ldots, w_k)$ in Γ with $w_1 = y$ (in this situation, we also say that y *has (at least)* $k-1$ *successor generations*).

(4) We say that Γ has *rigid procreation* if for all $y, z \in V$ and all positive integers k with $\mathrm{proc}_k(y), \mathrm{proc}_k(z) > 0$, one has $\mathrm{proc}_k(y) = \mathrm{proc}_k(z)$.

Using the notation of Definition 2.1.7, we note that $\mathrm{proc}_1(x)$ is simply the number of all children of x (i.e., the out-degree of x), that $\mathrm{proc}_2(x)$ is the number of children of x that have children themselves, etc. Rigid procreation means that all vertices with children must have the same number of children (though it is fine for vertices without children to exist), that all vertices with at least one "grandchild" must have the same number of children that have a child (in particular the same number of grandchildren), etc. The following fact was noted but not proved in [13, Remark before Theorem 3], and we prove it here (after the proof of Lemma 2.1.10) for the reader's convenience.

Proposition 2.1.8. *Let Γ be a finite digraph that is a functional graph (i.e., all vertices of Γ have out-degree 1), say $\Gamma = \Gamma_g$. If the dual digraph Γ^* has rigid procreation, then for any two g-periodic vertices x and y, we have $\mathrm{Tree}_\Gamma(x) \cong \mathrm{Tree}_\Gamma(y)$. Moreover, the common rooted tree isomorphism type above periodic vertices x in Γ is determined by the procreation number sequence $(\mathrm{proc}_k(x))_{k\geq 1}$ alone (i.e., it is the same in any finite functional graph with rigid procreation and the same procreation number sequence of periodic vertices).*

Readers interested in how the isomorphism type of $\mathrm{Tree}_\Gamma(x)$ can be derived from $(\mathrm{proc}_k(x))_{k\geq 1}$ for periodic vertices x if Γ^* has rigid procreation can find the details of this in Section 4.1. Before proving Proposition 2.1.8, we extend the notation $\mathrm{Tree}_\Gamma(x)$, which was already defined for finite functional graphs Γ in Chapter 1, to

the case where Γ is a finite directed rooted tree Δ (with all arcs oriented toward the root) and prove a lemma.

Definition 2.1.9. Let Δ be a finite directed rooted tree, with root $\mathrm{rt}(\Delta)$. We observe that all vertices except $\mathrm{rt}(\Delta)$ have out-degree 1, and we let g be the unique function $\mathrm{V}(\Delta) \setminus \{\mathrm{rt}(\Delta)\} \to \mathrm{V}(\Delta)$ such that Δ has an arc $x \to g(x)$ for each vertex $x \neq \mathrm{rt}(\Delta)$. For each $x \in \mathrm{V}(\Delta)$, we define $\mathrm{Tree}_\Delta(x)$, the so-called *tree above x in Δ*, as the induced subgraph of Δ on the set

$$\{x\} \cup \{y \in \mathrm{V}(\Delta) : y \neq x, \text{ and } g^k(y) = x \text{ for some } k = k(y) \geq 1\}.$$

In the statement of the following lemma and beyond, we denote the height of a finite directed rooted tree Δ by $\mathrm{ht}(\Delta)$.

Lemma 2.1.10. *Let Δ_1 and Δ_2 be finite directed rooted trees. Moreover, we assume that Δ_1 and Δ_2 have the same height, that the dual digraphs Δ_1^* and Δ_2^* both have rigid procreation, and that $\mathrm{proc}_h^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)) = \mathrm{proc}_h^{(\Delta_2^*)}(\mathrm{rt}(\Delta_2))$ for $1 \leq h \leq \mathrm{ht}(\Delta_1) = \mathrm{ht}(\Delta_2)$. Then Δ_1 and Δ_2 are isomorphic.*

Proof. We proceed by induction on the common height of Δ_1 and Δ_2. If $\mathrm{ht}(\Delta_1)$=0, then both Δ_1 and Δ_2 consist of a single vertex without any arcs and thus are isomorphic. Now we assume that $\mathrm{ht}(\Delta_1) \geq 1$ and that the statement holds for all smaller heights. We note that Δ_1 and Δ_2 are isomorphic if and only if the following equality of multisets holds, where $[\Gamma]_\cong$ denotes the isomorphism type of the finite digraph Γ:

$$\begin{aligned} &\{[\mathrm{Tree}_{\Delta_1}(y_1)]_\cong : y_1 \text{ is a child of } \mathrm{rt}(\Delta_1) \text{ in } \Delta_1^*\} \\ &\quad = \{[\mathrm{Tree}_{\Delta_2}(y_2)]_\cong : y_2 \text{ is a child of } \mathrm{rt}(\Delta_2) \text{ in } \Delta_2^*\}. \end{aligned} \tag{2.1}$$

It is thus our goal to prove equality (2.1). Let us fix $h \in \{0, 1, 2, \ldots, \mathrm{ht}(\Delta_1) - 1\}$. The number of children y_1 of $\mathrm{rt}(\Delta_1)$ in Δ_1^* such that $\mathrm{Tree}_{\Delta_1}(y_1)$ has height exactly h is

$$\mathrm{proc}_{h+1}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)) - \mathrm{proc}_{h+2}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)).$$

As for children y_2 of $\mathrm{rt}(\Delta_2)$ in Δ_2^* such that $\mathrm{Tree}_{\Delta_2}(y_2)$ has height exactly h, one finds analogously that their number is

$$\mathrm{proc}_{h+1}^{(\Delta_2^*)}(\mathrm{rt}(\Delta_2)) - \mathrm{proc}_{h+2}^{(\Delta_2^*)}(\mathrm{rt}(\Delta_2)) = \mathrm{proc}_{h+1}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)) - \mathrm{proc}_{h+2}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)).$$

Moreover, for each child y_1 of $\mathrm{rt}(\Delta_1)$ in Δ_1^* such that $\mathrm{Tree}_{\Delta_1}(y_1)$ has height exactly h, the first h procreation numbers of y_1 in $(\mathrm{Tree}_{\Delta_1}(y_1))^*$ are the same as those of $\mathrm{rt}(\Delta_1)$ in Δ_1^*, since Δ_1^* has rigid procreation. In particular, by the induction hypothesis and for fixed h, all digraphs $\mathrm{Tree}_{\Delta_1}(y_1)$, where y_1 is a child of $\mathrm{rt}(\Delta_1)$ in Δ_1^* with exactly h successor generations in Δ_1^* are isomorphic, their isomorphism type $\mathfrak{I}_h^{(1)}$ being determined by the first h procreation numbers of $\mathrm{rt}(\Delta_1)$ in Δ_1^*. An analogous statement

holds with Δ_2 and $\mathrm{rt}(\Delta_2)$ in place of Δ_1 and $\mathrm{rt}(\Delta_1)$, say with isomorphism type $\mathfrak{T}_h^{(2)}$. But by assumption, the procreation numbers of $\mathrm{rt}(\Delta_1)$ in Δ_1^* and of $\mathrm{rt}(\Delta_2)$ in Δ_2^* are the same, whence $\mathfrak{T}_h^{(1)} = \mathfrak{T}_h^{(2)}$ for each h. This shows that the two multisets in formula (2.1) are the same, each consisting of exactly $\mathrm{proc}_{h+1}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)) - \mathrm{proc}_{h+2}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1))$ copies of $\mathfrak{T}_h^{(1)}$ for each $h = 0, 1, \ldots, \mathrm{ht}(\Delta_1) - 1$. ∎

Proof of Proposition 2.1.8. First of all, in order for the assertion to make sense, we observe that the sequence $(\mathrm{proc}_k^{(\Gamma^*)}(x))_{k\geq 1}$ does not depend on the choice of periodic vertex x. Indeed, if y is another periodic vertex, then it is possible to form arbitrarily long directed paths in Γ^* starting at either of x or y by going along the respective cycle. This implies that $\mathrm{proc}_k^{(\Gamma^*)}(x), \mathrm{proc}_k^{(\Gamma^*)}(y) > 0$ for each k, and thus $\mathrm{proc}_k^{(\Gamma^*)}(x) = \mathrm{proc}_k^{(\Gamma^*)}(y)$ since Γ^* has rigid procreation.

Let us write $\Gamma = \Gamma_g$ for a suitably chosen function $g : \mathrm{V}(\Gamma) \to \mathrm{V}(\Gamma)$, and fix a g-periodic vertex x. We note that $\mathrm{Tree}_\Gamma(y) = \mathrm{Tree}_{\mathrm{Tree}_\Gamma(x)}(y)$ for each $y \in \mathrm{V}(\mathrm{Tree}_\Gamma(x))$. We observe that

$$\mathrm{proc}_k^{(\mathrm{Tree}_\Gamma(x)^*)}(x) = \mathrm{proc}_k^{(\Gamma^*)}(x) - 1$$

for each $k \geq 1$; this is because exactly one of the children of x counted by $\mathrm{proc}_k^{(\Gamma^*)}(x)$ is periodic and hence must be ignored in the procreation number in $\mathrm{Tree}_\Gamma(x)^*$. In particular, for each $h \geq 0$, the number of children y of x in $\mathrm{Tree}_\Gamma(x)^*$ such that $\mathrm{Tree}_\Gamma(y)$ has height exactly h is

$$\mathrm{proc}_{h+1}^{(\mathrm{Tree}_\Gamma(x)^*)}(x) - \mathrm{proc}_{h+2}^{(\mathrm{Tree}_\Gamma(x)^*)}(x) = \mathrm{proc}_{h+1}^{(\Gamma^*)}(x) - \mathrm{proc}_{h+2}^{(\Gamma^*)}(x).$$

Moreover, for each child y of x in $\mathrm{Tree}_\Gamma(x)^*$, the fact that Γ^* has rigid procreation implies that $\mathrm{Tree}_\Gamma(y)^*$ has rigid procreation, and, more specifically, whenever $\mathrm{proc}_k^{(\mathrm{Tree}_\Gamma(y)^*)}(z) > 0$ for some $z \in \mathrm{V}(\mathrm{Tree}_\Gamma(y))$, one has $\mathrm{proc}_k^{(\mathrm{Tree}_\Gamma(y)^*)}(z) = \mathrm{proc}_k^{\Gamma^*}(z) = \mathrm{proc}_k^{\Gamma^*}(x)$. Lemma 2.1.10 thus implies that the multiset of isomorphism types

$$\{[\mathrm{Tree}_\Gamma(y)]_\cong : g(y) = x,\ y \text{ is } g\text{-transient}\},$$

which determines the isomorphism type of $\mathrm{Tree}_\Gamma(x)$, is in turn entirely determined by the procreation number sequence $(\mathrm{proc}_k^{(\Gamma^*)}(x))_{k\geq 1}$, which is what we needed to prove. ∎

In view of Proposition 2.1.8, the following result, which is [13, Theorem 2], both implies that the rooted trees above periodic vertices in Γ_{μ_a}, the functional graph of the endomorphism $x \mapsto ax$ of $\mathbb{Z}/m\mathbb{Z}$, are pairwise isomorphic, and characterizes the corresponding rooted tree isomorphism type.

Theorem 2.1.11. *Let m be a positive integer, and let $\mu_a : x \mapsto ax$, be an endomorphism of the cyclic group $\mathbb{Z}/m\mathbb{Z}$. The dual functional graph $(\Gamma_{\mu_a})^*$ has rigid procreation, and for each $k \in \mathbb{N}^+$ and each periodic vertex x of Γ_{μ_a}, one has $\mathrm{proc}_k(x) = |\ker^{(k)}(\mu_a) : \ker^{(k-1)}(\mu_a)|$, where $\ker^{(j)}(\mu_a) := \{y \in \mathbb{Z}/m\mathbb{Z} : (\mu_a)^j(y) = a^j y = 0\}$.*

Recall that our goal is to understand the functional graphs of affine maps $A : x \mapsto ax + b$ of finite cyclic groups in general. Understanding that for $b = 0$, the graph has rigid procreation with known procreation numbers may only seem like a small special case. However, as we will see, both the rigid procreation behavior and the procreation numbers are preserved when passing to a general b (see Theorem 2.1.13). In fact, this is not even just a special property of finite cyclic groups but holds true for all finite groups (with a suitable definition of "affine map" – see Theorem 2.1.21 at the end of this section).

In view of Theorem 2.1.11, Theorem 2.1.5 is clear once we have proved the following result, which allows us more generally to derive the isomorphism type of Γ_A from the isomorphism types of the functional graphs Γ_{A_j} of the reductions of A modulo the prime power factors $p_j^{v_j}$ of m.

Proposition 2.1.12. *Let $g_j : X_j \to X_j$ for $j = 1, 2$ be functions on finite sets.*

(1) *We have* $\mathrm{per}(g_1 \otimes g_2) = \mathrm{per}(g_1) \times \mathrm{per}(g_2)$. *In particular, if g_1 and g_2 each have precisely one periodic point, then so does $g_1 \otimes g_2$.*

(2) *We assume that the dual functional graphs $\Gamma_{g_1}^*$ and $\Gamma_{g_2}^*$ have rigid procreation. Then the dual functional graph*

$$\Gamma_{g_1 \otimes g_2}^* \cong (\Gamma_{g_1} \otimes \Gamma_{g_2})^* = (\Gamma_{g_1})^* \otimes (\Gamma_{g_2})^*$$

has rigid procreation, and if y_j is a periodic point of g_j for $j = 1, 2$, then $\vec{y} = (y_1, y_2)$ is a periodic point of $g_1 \otimes g_2$ and $\mathrm{proc}_k(\vec{y}) = \mathrm{proc}_k(y_1) \cdot \mathrm{proc}_k(y_2)$.

(3) *Let us denote by $\divideontimes$ the unique $\mathbb{Q}$-bilinear product of polynomials in $\mathbb{Q}[x_n : n \in \mathbb{N}^+]$ such that*

$$(x_1^{e_1} \cdots x_N^{e_N}) \divideontimes (x_1^{e'_1} \cdots x_{N'}^{e'_{N'}}) = \prod_{1 \le n \le N, 1 \le n' \le N'} x_n^{e_n} \divideontimes x_{n'}^{e_{n'}}$$

and

$$x_n^e \divideontimes x_{n'}^{e'} = x_{\mathrm{lcm}(n,n')}^{ee' \gcd(n,n')}.$$

We assume that each g_j is a permutation of the respective set X_j. Then $g_1 \otimes g_2$ is a permutation of $X_1 \times X_2$, and its cycle type can be computed as the $\divideontimes$-product of the cycle types of g_1 and g_2.

(4) *We assume that g_1 has precisely one periodic point y_1 (a fixed point, necessarily) and that g_2 is a permutation of X_2. Then the induced subgraph of $g_1 \otimes g_2$ on* $\mathrm{per}(g_1 \otimes g_2)$ *is isomorphic to Γ_{g_2}, and for each $\vec{y} \in \mathrm{per}(g_1 \otimes g_2)$, one has* $\mathrm{Tree}_{\Gamma_{g_1 \otimes g_2}}(\vec{y}) \cong \mathrm{Tree}_{\Gamma_{g_1}}(y_1)$.

Proof. Since $g_1 \otimes g_2$ is the component-wise application of g_1 and g_2 on $X_1 \times X_2$, it is clear that a point $(y_1, y_2) \in X_1 \times X_2$ is periodic under $g_1 \otimes g_2$ if and only if y_j is

periodic under g_j for $j = 1, 2$, which settles statement (1) as well as the first assertion on $\vec{y}$ in statement (2).

For the rest of statement (2), we proceed as follows. In order to see that $(\Gamma_{g_1})^* \otimes (\Gamma_{g_2})^*$ has rigid procreation, let k be a positive integer, and let $\vec{y} = (y_1, y_2)$ and $\vec{z} = (z_1, z_2)$ be points in $X_1 \times X_2$ which have at least k successor generations each in that graph. This is equivalent to each of y_1, y_2, z_1, z_2 having at least k successor generations in the respective graph $\Gamma^*_{g_1}$ or $\Gamma^*_{g_2}$. It follows that $\operatorname{proc}_k(y_1) = \operatorname{proc}_k(z_1)$ and $\operatorname{proc}_k(y_2) = \operatorname{proc}_k(z_2)$. Now, for $\vec{w} = (w_1, w_2) \in X_1 \times X_2$, the procreation number $\operatorname{proc}_k(\vec{w})$ counts the number of children $\vec{w'} = (w_1', w_2')$ of $\vec{w}$ in $(\Gamma_{g_1})^* \otimes (\Gamma_{g_2})^*$ that have at least $k-1$ successor generations. But $\vec{w'}$ has at least $k-1$ successor generations if and only if each w_j' is a child of w_j in $\Gamma^*_{g_j}$ that has at least $k-1$ successor generations. Therefore, $\operatorname{proc}_k(\vec{w}) = \operatorname{proc}_k(w_1) \cdot \operatorname{proc}_k(w_2)$. In particular,

$$\operatorname{proc}_k(\vec{y}) = \operatorname{proc}_k(y_1)\operatorname{proc}_k(y_2) = \operatorname{proc}_k(z_1)\operatorname{proc}_k(z_2) = \operatorname{proc}_k(\vec{z}),$$

as required.

For statement (3), see [86, Theorem 2.4 and its proof].

For statement (4), we observe that this is implicit in [13, Theorem 1 (4)], but since it is not proved in detail there, let us do so here. By statement (1), we know that $\vec{z} = (z_1, z_2) \in X_1 \times X_2$ is a periodic point of $g_1 \otimes g_2$ if and only if $z_j \in \operatorname{per}(g_j)$ for $j = 1, 2$. Hence, the periodic points of $g_1 \otimes g_2$ are in bijection with those of g_2 via $y_2 \mapsto (y_1, y_2)$, and this bijection preserves cycle lengths. Therefore, the asserted isomorphism $\Gamma_{g_2} \cong \Gamma_{(g_1 \otimes g_2)_{|\operatorname{per}(g_1 \otimes g_2)}}$ is clear. Finally, it is not hard to check that for each $y_2 \in \operatorname{per}(g_2)$, the function $X_1 \to X_1 \times X_2$, $z_1 \mapsto (z_1, ((g_2)_{|\operatorname{per}(g_2)})^{-\operatorname{pperl}_{g_1}(z_1)}(y_2))$, is a digraph isomorphism between $\operatorname{Tree}_{\Gamma_{g_1}}(y_1)$ and $\operatorname{Tree}_{\Gamma_{g_1 \otimes g_2}}((y_1, y_2))$. ∎

We are now ready to prove Theorem 2.1.5. In fact, we prove the following stronger version of it.

Theorem 2.1.13. *Let m be a positive integer, and let $A : x \mapsto ax + b$ be an affine map of the cyclic group $\mathbb{Z}/m\mathbb{Z}$. The dual functional graph $(\Gamma_A)^*$ has rigid procreation, and for each $k \in \mathbb{N}^+$ and each periodic vertex x of Γ_A, one has $\operatorname{proc}_k(x) = |\ker^{(k)}(\mu_a) : \ker^{(k-1)}(\mu_a)|$. In particular, each rooted tree above a periodic vertex in Γ_A is isomorphic to each rooted tree above a periodic vertex in Γ_{μ_a}.*

We note that in the situation of Theorem 2.1.13, one has $|\ker^{(j)}(\mu_a)| = \gcd(a^j, m)$ for all $j \in \mathbb{N}_0$, making the computation of the procreation numbers (and thus the understanding of the isomorphism types of rooted trees above periodic vertices) easy.

Proof of Theorem 2.1.13. The "In particular" statement is clear by Proposition 2.1.8 and Theorem 2.1.11, so we focus on the proof of the main statement.

As above, we factor $m = p_1^{v_1} \cdots p_K^{v_K}$ and denote by A_j the reduction of A modulo $p_j^{v_j}$. We claim that for each $j = 1, 2, \ldots, K$, the dual functional graph $(\Gamma_{A_j})^*$ has

rigid procreation, and that for each A_j-periodic vertex $x \in \mathbb{Z}/p_j^{v_j}\mathbb{Z}$, the procreation number sequence of x in $(\Gamma_{A_j})^*$ agrees with that of a periodic vertex in the dual functional graph of the reduction $(\mu_a)_j$ of μ_a modulo $p_j^{v_j}$. Indeed, following the proof of Lemma 2.1.6, this is clear both if $\nu_{p_j}^{(v_j)}(a-1) \leq \nu_{p_j}^{(v_j)}(b)$ (where the digraphs $(\Gamma_{A_j})^*$ and $(\Gamma_{(\mu_a)_j})^*$ are actually isomorphic as a whole) and if $\nu_{p_j}^{(v_j)}(a-1) > \nu_{p_j}^{(v_j)}(b)$ (where both $(\Gamma_{A_j})^*$ and $(\Gamma_{(\mu_a)_j})^*$ are disjoint unions of directed cycles).

Now, we recall that $\Gamma_A = \bigotimes_{j=1}^K \Gamma_{A_j}$ and $\Gamma_{\mu_a} = \bigotimes_{j=1}^K \Gamma_{(\mu_a)_j}$. In both tensor products, the duals of the factors indexed by j have rigid procreation with the same procreation number sequences of periodic vertices. Therefore, by Proposition 2.1.12 (2), the same applies to $(\Gamma_A)^*$ versus $(\Gamma_{\mu_a})^*$, and this settles the main statement by virtue of the formula for procreation numbers in Theorem 2.1.11. ∎

We also note the following consequence of Proposition 2.1.12 (1), which will become important later.

Lemma 2.1.14. *Let m be a positive integer, let $a, b \in \mathbb{Z}/m\mathbb{Z}$, and let L be a positive integer that is so large that $\gcd(a^L, m) = \prod_{p \mid \gcd(a,m)} p^{\nu_p(m)}$ (for example, $\mathrm{mpe}(m) := \max_p \nu_p(m) \leq \lfloor \log_2 m \rfloor$ is a valid choice for L). Then $y \in \mathbb{Z}/m\mathbb{Z}$ is a periodic point of the affine map $A : z \mapsto az + b$ of $\mathbb{Z}/m\mathbb{Z}$ if and only if*

$$y \equiv \sum_{t=0}^{L-1} a^t \cdot b \pmod{\gcd(a^L, m)}.$$

We take note of a notation used especially in group-theoretic literature that can be used to denote the product $\prod_{p \mid \gcd(a,m)} p^{\nu_p(m)}$ from Lemma 2.1.14 in a compact form. Namely, for a set P of primes and a positive integer n, the *P-part of n*, written n_P, is defined as $\prod_{p \in P} p^{\nu_p(n)}$. Accordingly, the product $\prod_{p \mid \gcd(a,m)} p^{\nu_p(m)}$ may be written as $m_{\pi(a)}$, where $\pi(a)$ denotes the set of prime divisors of a.

Proof of Lemma 2.1.14. By Proposition 2.1.12 (1) and the Chinese remainder theorem, a point $y \in \mathbb{Z}/m\mathbb{Z}$ is periodic under A if and only if the reduction y_p of y modulo $p^{\nu_p(m)}$ is periodic under the reduction A_p of A modulo $p^{\nu_p(m)}$ for each prime $p \mid m$. But if $p \nmid a$, then by Proposition 2.1.4 (2), A_p is a permutation of $\mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$, so y_p is periodic under A_p. Therefore, only the primes $p \mid \gcd(a, m)$ actually give a restriction on y. For those primes p, we know by Proposition 2.1.4 (1) that A_p has precisely one periodic point. It follows that if $y_0 \in \mathbb{Z}/m\mathbb{Z}$ is a periodic point of A, then the periodic points y of A are characterized by the congruence $y \equiv y_0 \pmod{\gcd(a^L, m)}$. Therefore, it only remains to prove that $\sum_{t=0}^{L-1} a^t \cdot b$ is a periodic point of A.

Now, using the same argument with $b = 0$ so that it applies to $\mu_a : z \mapsto az$, we see that periodic points y of μ_a are characterized by the congruence

$$y \equiv 0 \pmod{\gcd(a^L, m)}.$$

In particular, $(\mu_a)^L(z) = a^L z$ is periodic under μ_a for each $z \in \mathbb{Z}/m\mathbb{Z}$. Since the trees above periodic vertices in the functional graphs of μ_a and A are isomorphic (by Theorem 2.1.5) and thus have the same height, it follows that $A^L(z)$ is periodic under A for each $z \in \mathbb{Z}/m\mathbb{Z}$. In particular, $A^L(0) = \sum_{t=0}^{L-1} a^t \cdot b$ is periodic under A, as we needed to show. ∎

This concludes our results for cyclic groups. To round this section off, we put in some extra work to generalize Theorem 2.1.13 to arbitrary finite groups. First, we need to clarify what we mean by "affine map" in general. In what follows, for a fixed group G we denote by $\rho_{\mathrm{r}} : G \to \mathrm{Sym}(G)$, $x \mapsto (y \mapsto yx)$, the so-called *right-regular representation of G on itself* (for each $x \in G$, the function value $\rho_{\mathrm{r}}(x) \in \mathrm{Sym}(G)$ is the right-multiplication by x on G). Analogously, ρ_{l} denotes the *left-regular representation of G on itself*, whose function values are the left-multiplications on G by fixed elements of G. As is common in group theory, we write the composition of a function $g : X \to Y$ with a function $g' : Y \to Z$ as the product $gg' : X \to Z$ (a synonymous notation is $g' \circ g$). When using this notation for composition, applications of functions to arguments are commonly written using exponents: x^g instead of $g(x)$, so that $x^{gg'} = (x^g)^{g'}$.

Definition 2.1.15. Let G be a group. An *affine map of G* is a function $G \to G$ of the form $\varphi\rho_{\mathrm{r}}(b) : x \mapsto x^\varphi b$ for some fixed element $b \in G$ and group endomorphism φ of G.

Remark 2.1.16. We note the following concerning the concept of an affine map.

(1) Since $\varphi\rho_{\mathrm{r}}(1_G) = \varphi$, affine maps are generalizations of group endomorphisms.

(2) The affine maps of a given group G form a monoid of functions on G, as they are composed via the formula

$$\varphi\rho_{\mathrm{r}}(b) \cdot \varphi'\rho_{\mathrm{r}}(b') = \varphi\varphi'\rho_{\mathrm{r}}(b^{\varphi'}b').$$

(3) Alternatively, one could define affine maps through left-multiplication by a constant after application of a group endomorphism φ, i.e., as compositions $\varphi\rho_{\mathrm{l}}(b) = \rho_{\mathrm{l}}(b) \circ \varphi$. This leads to the same class of functions, as

$$(\rho_{\mathrm{l}}(b) \circ \varphi)(x) = b\varphi(x) = bx^\varphi = bx^\varphi b^{-1}b = x^{\varphi\,\mathrm{conj}(b^{-1})}b = x^{\varphi\,\mathrm{conj}(b^{-1})\cdot\rho_{\mathrm{r}}(b)},$$

where $\mathrm{conj}(g)$ is the inner automorphism (conjugation) $x \mapsto g^{-1}xg$ of G.

We briefly review some more concepts and results from [13].

Definition 2.1.17. Let G be a finite group and φ an endomorphism of G. The *hyperkernel of φ*, written $\mathrm{nil}(\varphi)$, consists of those $x \in G$ such that $\varphi^n(x) = 1_G$ for some $n = n(x) \in \mathbb{N}_0$.

Remark 2.1.18. The notation $\mathrm{nil}(\varphi)$ stems from the fact that this is the largest φ-invariant subgroup of G of which the corresponding restriction of φ is a nilpotent endomorphism (i.e., an endomorphism that becomes trivial if composed with itself sufficiently often). This subgroup was called the *nilpotent part of G with respect to φ* in [13], and the authors are not aware of any other names used for it earlier. In particular, in Caranti's paper [16], to which Theorem 2.1.19 below can be traced back, no name was given to this subgroup. The authors decided to switch to the terminology "hyperkernel" in this memoir instead because calling $\mathrm{nil}(\varphi)$ "nilpotent part" may give the wrong impression that it is always a nilpotent group.

In the following theorem, we use the notation $G = H \ltimes N$ to express that the group G is the *(internal) semidirect product of H and N*, which means that H is a subgroup of G, that N is a normal subgroup of G, and that one has $H \cap N = \{1_G\}$ and $HN = \{hn : h \in H, n \in N\} = G$. In this situation, each element of G can be written as a product hn for $h \in H$ and $n \in N$ *in a unique way*, and one may multiply elements of G via the formula $(hn) \cdot (h'n') = hh'n^{h'}n'$, where $n^{h'} = n^{\mathrm{conj}(h')} = (h')^{-1}nh'$.

Theorem 2.1.19. *Let G be a finite group, and φ an endomorphism of G. Then $G = \mathrm{per}(\varphi) \ltimes \mathrm{nil}(\varphi)$.*

Proof. See [13, proof of Theorem 1 (1–3)]. We note that after observing that $\mathrm{per}(\varphi)$ is a subgroup and $\mathrm{nil}(\varphi)$ is a normal subgroup of G, the rest of the statement follows easily from a group-theoretic version of Fitting's lemma that was proved by Caranti, see [16, Theorem 4.2]. ∎

Theorem 2.1.19 has an interesting consequence concerning the functional graph Γ_φ, which was originally stated as [13, Theorem 1 (4)] and is easy to prove using Proposition 2.1.12 with $X_1 = \mathrm{per}(\varphi)$ and $X_2 = \mathrm{nil}(\varphi)$.

Corollary 2.1.20. *Let G be a finite group, and φ an endomorphism of G. Then*

$$\Gamma_\varphi = \Gamma_{\varphi|_{\mathrm{per}(\varphi)}} \otimes \Gamma_{\varphi|_{\mathrm{nil}(\varphi)}},$$

and, consequently, for each $x \in \mathrm{per}(\varphi)$, one has $\mathrm{Tree}_{\Gamma_\varphi}(x) \cong \mathrm{Tree}_{\Gamma_{\varphi|_{\mathrm{nil}(\varphi)}}}(1_G)$, a rooted tree that can be obtained from $\Gamma_{\varphi|_{\mathrm{nil}(\varphi)}}$ by deleting the unique loop of the latter at 1_G.

The following result extends Theorem 2.1.13 to arbitrary finite groups.

Theorem 2.1.21. *Let G be a finite group, $b \in G$, and φ an endomorphism of G. Then $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ has rigid procreation. Moreover, the sequence of procreation numbers $(\mathrm{proc}_k(x))_{k\geq 1}$ of any periodic vertex x in $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ is the same as that of a periodic vertex in Γ^*_φ.*

Proof. We prove by induction on $k \geq 1$ that if $x, y \in G$ each have at least k successor generations in $\Gamma^*_{\varphi\rho_r(b)}$, then $\mathrm{proc}_k(x) = \mathrm{proc}_k(y)$ in $\Gamma^*_{\varphi\rho_r(b)}$. For the induction base, $k = 1$, we observe that for $z \in \{x, y\}$, one has

$$\begin{aligned}\mathrm{proc}_1(z) &= \#\text{ children of } z \text{ in } \Gamma^*_{\varphi\rho_r(b)} \\ &= |\{w \in G : w^{\varphi\rho_r(b)} = z\}| = |\{w \in G : w^{\varphi} = zb^{-1}\}|.\end{aligned}$$

Since $\{w \in G : w^{\varphi} = zb^{-1}\}$ is either empty or a coset of $\ker\varphi$, it follows that $\mathrm{proc}_1(x) = \mathrm{proc}_1(y) = |\ker\varphi|$ whenever $\mathrm{proc}_1(x), \mathrm{proc}_1(y) > 0$, as required.

Now we assume that $k \geq 2$ and that the statement holds up to $k - 1$. Each of the two vertices x and y has at least n successor generations in $\Gamma^*_{\varphi\rho_r(b)}$ for each $n \in \{1, 2, \ldots, k-1\}$, whence by the induction hypothesis, one has $\mathrm{proc}_n(x) = \mathrm{proc}_n(y)$ for $n = 1, 2, \ldots, k-1$. Now, for each $z \in G$, the number of endpoints of directed paths of length k in $\Gamma^*_{\varphi\rho_r(b)}$ starting at z is

$$\begin{aligned}|\{w \in G : w^{((\varphi\rho_r(b))^k)} = z\}| &= |\{w \in G : \varphi^k(w) \cdot \varphi^{k-1}(b)\varphi^{k-2}(b) \cdots \varphi(b)b = z\}| \\ &= |\{w \in G : \varphi^k(w) = zb^{-1}\varphi(b)^{-1} \cdots (\varphi^{k-1}(b))^{-1}\}|,\end{aligned}$$

and the set $\{w \in G : \varphi^k(w) = zb^{-1}\varphi(b)^{-1} \cdots (\varphi^{k-1}(b))^{-1}\}$ is either empty or a coset of

$$\ker^{(k)}(\varphi) := \{w \in G : \varphi^k(w) = 1_G\}.$$

Hence, x and y have the same number of endpoints of length k directed paths in $\Gamma^*_{\varphi\rho_r(b)}$ starting at them, namely $|\ker^{(k)}(\varphi)|$. Using the induction hypothesis in its general form (which basically states that $\Gamma^*_{\varphi\rho_r(b)}$ has rigid procreation "for $k-1$ generations"), an easy induction on $n = 1, 2, \ldots, k-1$ shows that for each vertex $w \in G$ with at least n successor generations in $\Gamma^*_{\varphi\rho_r(b)}$, the number of endpoints of directed paths in $\Gamma^*_{\varphi\rho_r(b)}$ of length n starting at w is $\prod_{j=1}^{n} \mathrm{proc}_j(w)$. In particular, if w is a child of $z \in \{x, y\}$ that has at least $k-1$ successor generations, then the number of endpoints of length $k-1$ paths starting at w is $\prod_{j=1}^{k-1} \mathrm{proc}_j(w)$, which is equal to $\prod_{j=1}^{k-1} \mathrm{proc}_j(z)$ by the induction hypothesis. Since the number of such children w of z is $\mathrm{proc}_k(z)$, we conclude that

$$\begin{aligned}\prod_{j=1}^{k} \mathrm{proc}_j(z) &= \mathrm{proc}_k(z) \cdot \prod_{j=1}^{k-1} \mathrm{proc}_j(z) \\ &= \#\text{ endpoints of length } k \text{ paths starting at } z = |\ker^{(k)}(\varphi)|.\end{aligned}$$

Because $z \in \{x, y\}$ is arbitrary, it follows that

$$\prod_{j=1}^{k} \mathrm{proc}_j(x) = \prod_{j=1}^{k} \mathrm{proc}_j(y),$$

and since $\mathrm{proc}_j(x) = \mathrm{proc}_j(y) > 0$ for $j = 1, 2, \ldots, k-1$, this allows us to conclude that $\mathrm{proc}_k(x) = \mathrm{proc}_k(y)$, as required.

Concerning the claim that the procreation numbers of $\Gamma^*_{\varphi\rho_r(b)}$ and Γ^*_φ are the same, the above argument shows that for any positive integer k and any periodic vertex x of $\Gamma^*_{\varphi\rho_r(b)}$, we have

$$\prod_{j=1}^{k} \mathrm{proc}_j(x) = |\ker^{(k)}(\varphi)|.$$

It follows that

$$\mathrm{proc}_k(x) = |\ker^{(k)}(\varphi) : \ker^{(k-1)}(\varphi)|$$

for each $k \in \mathbb{N}^+$ (we note that $\ker^{(0)}(\varphi) = \{1_G\}$). Therefore, the procreation number sequence of a periodic vertex of $\Gamma^*_{\varphi\rho_r(b)}$ only depends on φ, not on b, and for $b := 1_G$, one has $\Gamma^*_{\varphi\rho_r(b)} = \Gamma^*_\varphi$. ∎

2.2 The master lemma

From the introduction, we recall the notion of an m-CC (short for "m-congruential condition"), which we defined as a condition of the form $\nu(x \equiv \mathfrak{b} \pmod{\mathfrak{a}})$ with $\mathfrak{a} \mid m$ and $\nu \in \{\emptyset, \neg\}$. In this section, we consider systems formed from m-CCs in one common variable. Such a system is *consistent* if it has an integer solution, and two such systems are *equivalent* if they have the same solution set in $\mathbb{Z}$ (or, equivalently, in $\mathbb{Z}/m\mathbb{Z}$). The solution set in $\mathbb{Z}/m\mathbb{Z}$ of a consistent system of m-CCs is a block of the associated arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ (see Definition 1.5). We note the following fundamental result on systems of m-congruences, which is a well-known generalization of the Chinese remainder theorem.

Proposition 2.2.1. *Let m be a positive integer. We consider a system of m-congruences, of the form*

$$\begin{aligned} x &\equiv \mathfrak{b}_1 \pmod{\mathfrak{a}_1}, \\ x &\equiv \mathfrak{b}_2 \pmod{\mathfrak{a}_2}, \\ &\vdots \\ x &\equiv \mathfrak{b}_K \pmod{\mathfrak{a}_K}. \end{aligned} \tag{2.2}$$

The following statements are equivalent.

(1) *System (2.2) is consistent.*

(2) *For all $1 \le j < k \le K$, one has $\gcd(\mathfrak{a}_j, \mathfrak{a}_k) \mid \mathfrak{b}_j - \mathfrak{b}_k$.*

(3) *Any pair of m-congruences in system (2.2) form a consistent system.*

(4) *System* (2.2) *is equivalent to a single m-congruence, of the form*

$$x \equiv \mathfrak{b} \pmod{\operatorname{lcm}(\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_K)}.$$

In particular, if system (2.2) *is consistent, then its number of solutions modulo m is equal to* $m/\operatorname{lcm}(\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_K)$ *and is, therefore, independent of the* $\mathfrak{b}_j$.

Proof. For the equivalence "(1)⇔(2)" and the implication "(1)⇒(4)", see [41, Theorem 3.3.4 on p. 78], for example. Moreover, the implication "(4)⇒(1)" is trivial. As for the equivalence "(1)⇔(3)", we note that by the already established equivalence "(1)⇔(2)", applied to the system

$$\begin{aligned} x &\equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j}, \\ x &\equiv \mathfrak{b}_k \pmod{\mathfrak{a}_k}, \end{aligned} \tag{2.3}$$

that system is consistent if and only if the single divisibility $\gcd(\mathfrak{a}_j, \mathfrak{a}_k) \mid \mathfrak{b}_j - \mathfrak{b}_k$ holds. But statement (3) just demands that system (2.3) be consistent for all $1 \leq j < k \leq K$, which is therefore equivalent to statement (2), and thus to statement (1). ∎

Proposition 2.2.1 is the basis for proving the following lemma, which is crucial for our recursive approach for understanding the rooted trees in Section 3.3.

Lemma 2.2.2. *[Master lemma] Let m be a positive integer, and let* $\mathcal{P}$ *be an arithmetic partition of* $\mathbb{Z}/m\mathbb{Z}$. *Moreover, let* $a, b \in \mathbb{Z}/m\mathbb{Z}$, *and consider the affine map* $A : x \mapsto ax + b$ *of* $\mathbb{Z}/m\mathbb{Z}$. *There is an arithmetic partition* $\mathcal{P}'$ *of* $\mathbb{Z}/m\mathbb{Z}$ *with* $\mathrm{AC}(\mathcal{P}') \leq \mathrm{AC}(\mathcal{P}) + 1$ *such that if* $x, y \in \mathbb{Z}/m\mathbb{Z}$ *are from a common block of* $\mathcal{P}'$, *and if B is a block of* $\mathcal{P}$, *then* $|A^{-1}(\{x\}) \cap B| = |A^{-1}(\{y\}) \cap B|$.

More specifically, if $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$, *then*

$$\mathfrak{P}'(\mathcal{P}, A) := \mathfrak{P}\left(\begin{array}{l} x \equiv a\mathfrak{b}_1 + b \pmod{\gcd(a\mathfrak{a}_1, m)} \\ x \equiv a\mathfrak{b}_2 + b \pmod{\gcd(a\mathfrak{a}_2, m)} \\ \quad \vdots \\ x \equiv a\mathfrak{b}_K + b \pmod{\gcd(a\mathfrak{a}_K, m)} \\ x \equiv b \pmod{\gcd(a, m)} \end{array}\right)$$

is a valid choice for $\mathcal{P}'$, *and a formula for* $|A^{-1}(\{x\}) \cap B|$ *in terms of the* $\mathcal{P}$*-block B and the unique* $\mathfrak{P}'(\mathcal{P}, A)$*-block* B' *containing x can be obtained as follows: write* $B = \mathfrak{B}(\mathcal{P}, \vec{\nu})$ *and* $B' = \mathfrak{B}(\mathfrak{P}'(\mathcal{P}, A), \vec{\nu'})$ *with* $\vec{\nu} = (\nu_1, \ldots, \nu_K) \in \{\emptyset, \neg\}^K$ *and* $\vec{\nu'} = (\nu'_1, \ldots, \nu'_{K+1}) \in \{\emptyset, \neg\}^{K+1}$. *We introduce the following notations.*

- $J_-(\vec{\nu}) := \{j \in \{1, 2, \ldots, K\} : \nu_j = \neg\}$ *and (analogously)*

$$J_-(\vec{\nu'}) := \{j' \in \{1, 2, \ldots, K+1\} : \nu'_{j'} = \neg\};$$

- $J_+(\vec{\nu}) := \{1, 2, \ldots, K\} \setminus J_-(\vec{\nu})$;

- *For $J \subseteq J_-(\vec{\nu})$, we denote by $E(\vec{\nu}, J)$ the condition*

$$\gcd(\alpha_{j_1}, \alpha_{j_2}) \mid \mathfrak{b}_{j_1} - \mathfrak{b}_{j_2} \quad \textit{for all } j_1, j_2 \in J_+(\vec{\nu}) \cup J.$$

Then for each $x \in \mathcal{B}(\mathfrak{P}'(\mathcal{P}, A), \vec{\nu}')$, the intersection size $|A^{-1}(\{x\}) \cap \mathcal{B}(\mathcal{P}, \vec{\nu})|$ is equal to

$$\sum_{J \subseteq J_-(\vec{\nu})} (-1)^{|J|} \kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}', J) =: \sigma_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}'),$$

where

$$\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}', J) := \delta_{\nu'_{K+1}=\emptyset} \cdot \delta_{E(\vec{\nu},J)} \cdot \delta_{(J_+(\vec{\nu})\cup J)\cap J_-(\vec{\nu}')=\emptyset} \cdot \frac{m}{\operatorname{lcm}(\frac{m}{\gcd(a,m)}, \alpha_j : j \in J_+(\vec{\nu}) \cup J)},$$

the three deltas being Kronecker deltas.

We call a number of the form $\sigma_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}')$ a *distribution number of $\mathcal{P}$ (under A).*

Proof of Lemma 2.2.2. To verify that

$$\mathcal{P}' := \mathfrak{P}'(\mathcal{P}, A)$$

has the desired property, we set

- $M_j := \{x \in \mathbb{Z}/m\mathbb{Z} : x \equiv \mathfrak{b}_j \pmod{\alpha_j}\}$ for $j = 1, 2, \ldots, K$;
- $M_+(\vec{\nu}) := \bigcap_{j \in J_+(\vec{\nu})} M_j$;

and note that $B = M_+(\vec{\nu}) \cap \bigcap_{j \in J_-(\vec{\nu})} M_j^c$ (the superscript c denoting set complementation in $\mathbb{Z}/m\mathbb{Z}$). Therefore, by the inclusion-exclusion principle, the intersection size $|A^{-1}(\{x\}) \cap B|$ we are looking for is equal to

$$\sum_{J \subseteq J_-(\vec{\nu})} (-1)^{|J|} \Big| (M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j \in J} M_j \Big|.$$

It is thus our goal to argue that the value of this sum is equal to $\sigma_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}')$ (in particular, it is independent of the choice of $x \in B' = \mathcal{B}(\mathcal{P}', \vec{\nu}')$). We do so by arguing that, in fact, the intersection size

$$\Big| (M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j \in J} M_j \Big|$$

is equal to $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}', J)$, for each $J \subseteq J_-(\vec{\nu})$. Now, writing

$$J_+(\vec{\nu}) \cup J = \{j_1, j_2, \ldots, j_N\},$$

the intersection $(M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j\in J} M_j$ is the solution set in $\mathbb{Z}/m\mathbb{Z}$ of the following system in the variable y:

$$\begin{aligned} y &\equiv \mathfrak{b}_{j_1} \pmod{\mathfrak{a}_{j_1}},\\ y &\equiv \mathfrak{b}_{j_2} \pmod{\mathfrak{a}_{j_2}},\\ &\vdots\\ y &\equiv \mathfrak{b}_{j_N} \pmod{\mathfrak{a}_{j_N}},\\ ay + b &\equiv x \pmod{m}. \end{aligned} \tag{2.4}$$

Now, the single congruence $ay + b \equiv x \pmod{m}$ is solvable in y if and only if $\gcd(a,m) \mid x - b$, i.e., if and only if $x \equiv b \pmod{\gcd(a,m)}$. This is one of the spanning congruences for $\mathcal{P}'$, whence its truth value is constant for all $x \in B'$. If that congruence is false for all $x \in B'$ (equivalently, if $\nu'_{K+1} = \neg$), then system (2.4) is always false, whence $|(M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j\in J} M_j| = 0$ for all $J \subseteq J_-(\vec{\nu})$, independently of x. This explains the first Kronecker delta in the definition of $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'}, J)$.

We may thus henceforth assume that $\nu'_{K+1} = \emptyset$, i.e., that $x \equiv b \pmod{\gcd(a,m)}$ for all $x \in B'$. Then the congruence $ay + b \equiv x \pmod{m}$ can be equivalently rewritten into

$$y \equiv \operatorname{inv}_{m/\gcd(a,m)}\left(\frac{a}{\gcd(a,m)}\right) \cdot \frac{x-b}{\gcd(a,m)} \pmod{\frac{m}{\gcd(a,m)}}, \tag{2.5}$$

where $\operatorname{inv}_n(x)$ denotes the multiplicative inverse of the unit x modulo n. If we replace the congruence $ay + b \equiv x \pmod{m}$ in the system (2.4) by the equivalent congruence (2.5), then the resulting system consists entirely of m-congruences. By Proposition 2.2.1, there are only two possibilities for the number of solutions modulo m of this system: either the system is inconsistent and, thus, has 0 solutions, or it has m/L solutions, where L is the least common multiple of the moduli that occur. Neither of these two expressions for the number of solutions depends on x, so we aim to show that it does not depend on the choice of $x \in B'$ which of the two cases occurs.

Now, Proposition 2.2.1 also implies that system (2.4) is consistent if and only if any pair of conditions in it is consistent. It thus suffices to argue that for no pair of conditions in system (2.4) does the consistency of the system formed from those two conditions depend on the choice of $x \in B'$. If both of those conditions are distinct from the congruence $ay + b \equiv x \pmod{m}$, then they are of the forms $y \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j}$ and $y \equiv \mathfrak{b}_k \pmod{\mathfrak{a}_k}$, for suitable $j, k \in J_+(\vec{\nu}) \cup J$, and by Proposition 2.2.1, those two conditions form a consistent system if and only if $\gcd(\mathfrak{a}_j, \mathfrak{a}_k) \mid \mathfrak{b}_j - \mathfrak{b}_k$. Equivalently, the system obtained from (2.4) by deleting the single congruence $ay + b \equiv x \pmod{m}$ is consistent if and only if the condition $E(\vec{\nu}, J)$ holds, which explains the second Kronecker delta in the definition of $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'}, J)$.

It remains to consider two-condition subsystems of (2.4) of the form

$$\begin{aligned} ay + b &\equiv x \pmod{m}, \\ y &\equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} \end{aligned} \tag{2.6}$$

for some $j \in J_+(\vec{\nu}) \cup J$. We claim that system (2.6) is consistent if and only if $x \equiv a\mathfrak{b}_j + b \pmod{\gcd(a\mathfrak{a}_j, m)}$. Indeed, if system (2.6) is consistent, then there is a $k \in \mathbb{Z}$ such that some $y = \mathfrak{b}_j + k\mathfrak{a}_j$ satisfies the first condition of the system. That is, one then has

$$x \equiv ay + b = a\mathfrak{b}_j + ka\mathfrak{a}_j + b \pmod{m}.$$

In particular,

$$x \equiv a\mathfrak{b}_j + ka\mathfrak{a}_j + b \equiv a\mathfrak{b}_j + b \pmod{\gcd(a\mathfrak{a}_j, m)},$$

as required. On the other hand, let us assume that

$$x \equiv a\mathfrak{b}_j + b \pmod{\gcd(a\mathfrak{a}_j, m)}.$$

Then we can write $x = a\mathfrak{b}_j + b + k' \gcd(a\mathfrak{a}_j, m)$ for some $k' \in \mathbb{Z}$. We need to verify that there is an integer k such that for $y = \mathfrak{b}_j + k\mathfrak{a}_j$, one has

$$ay + b = a\mathfrak{b}_j + ka\mathfrak{a}_j + b \equiv x = a\mathfrak{b}_j + b + k' \gcd(a\mathfrak{a}_j, m) \pmod{m},$$

which is equivalent to $ka\mathfrak{a}_j \equiv k' \gcd(a\mathfrak{a}_j, m) \pmod{m}$. And indeed, this is solvable in k, because $\gcd(a\mathfrak{a}_j, m) \mid k' \gcd(a\mathfrak{a}_j, m)$.

Now, because $x \equiv a\mathfrak{b}_j + b \pmod{\gcd(a\mathfrak{a}_j, m)}$ is the j-th spanning congruence of $\mathcal{P}'$, it follows that if $\nu'_j = \neg$ (equivalently, if $j \in J_-(\vec{\nu}')$), then the intersection $(M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j \in J} M_j$ is empty whenever $j \in J_+(\vec{\nu}) \cup J$ as well, which explains the third Kronecker delta in the definition of $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}', J)$.

If all three Kronecker deltas in the definition of $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}', J)$ are 1, then our argumentation shows that system (2.4) is consistent and is thus equivalent to a single m-congruence by an application of Proposition 2.2.1 (we recall that the last congruence in system (2.4) may be replaced by the equivalent m-congruence (2.5)), the modulus of which is the least common multiple L of the moduli involved in the m-congruence forms of the conditions in system (2.4). It follows that the size of the solution set of system (2.4) then is

$$\frac{m}{L} = \frac{m}{\operatorname{lcm}\left(\frac{m}{\gcd(a,m)},\ \mathfrak{a}_j : j \in J_+(\vec{\nu}) \cup J\right)}.$$

Therefore, our technical parameter $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu}', J)$ indeed always agrees with the intersection size $|(M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j \in J} M_j|$, and this concludes the proof. ∎

2.3 CRL-lists of affine maps of finite cyclic groups

We view $\mathbb{Z}/m\mathbb{Z}$, where $m \in \mathbb{N}^+$, as a ring with underlying set $\{0, 1, \ldots, m-1\}$ and modular addition and modular multiplication as the ring operations. In particular, if $m_1 \leq m_2$ are positive integers, then we have an inclusion of sets $\mathbb{Z}/m_1\mathbb{Z} \subseteq \mathbb{Z}/m_2\mathbb{Z}$. We may also view integers outside of the range $\{0, 1, \ldots, m-1\}$ as elements of $\mathbb{Z}/m\mathbb{Z}$, via reduction modulo m (identifying $x \in \mathbb{Z}$ with $x \bmod m$). We remind the reader of the notation $\nu_p^{(v)}(x) := \min\{v, \nu_p(x)\}$ for p prime, $v \in \mathbb{N}_0$ and $x \in \mathbb{Z}$, originally introduced after Theorem 2.1.5.

As was mentioned in the introduction, the construction of a CRL-list (in the sense of Definition 1.2) for a generalized cyclotomic mapping of $\mathbb{F}_q$ can be reduced to the corresponding problem for affine maps of finite cyclic groups, which we solve in this section; see Section 3.1 for the application of this to constructing CRL-lists of generalized cyclotomic mappings. We also observed in the introduction that determining a CRL-list for a function $g : X \to X$ with X finite is generally a harder problem than the determination of the cycle type of $g_{|\mathrm{per}(g)}$, and we would like to give an overview of the history of the latter problem for the case where g is an affine permutation of a finite cyclic group.

Ahmad [5] determined the cycle structure of *automorphisms* of finite cyclic groups. The cycle index of the group of affine permutations of a finite cyclic group $\mathbb{Z}/m\mathbb{Z}$ (which is a polynomial that encodes how many affine permutations of each given cycle type there are) was described by Wei and Xu [86]. In their paper, they gave the formulas for the case where m is a prime power without proof, referring to the two-page research announcement [85] by Wei, Gao and Yang. Unfortunately, while Bors and Wang were working on [15], they were unable to find [85] through an online search, which led them to derive those formulas independently as [15, Theorem 4.8], based on a precise description of the cycle type of a given affine permutation $A : x \mapsto ax + b$ of $\mathbb{Z}/p^v\mathbb{Z}$ in terms of a and b, stated as [15, Proposition 4.7]. This latter result is as of now, to the authors' knowledge, the only accessible reference that lists those cycle types explicitly, in a tabular form. While working on the current memoir, the authors realized that [15, Proposition 4.7] could also have been easily derived from Deng's results [21, Lemmas 4 and 7] and Ahmad's result [5, Theorem 1].

Let us now turn to the determination of CRL-lists. Let m be a positive integer, and let $a, b \in \mathbb{Z}$. We consider the affine map $A : x \mapsto ax + b$ of $\mathbb{Z}/m\mathbb{Z}$. From Proposition 2.1.4, we know the following.

- The reduction of A modulo $m' := \prod_{p|m, p\nmid a} p^{\nu_p(m)}$ is an affine permutation of $\mathbb{Z}/m'\mathbb{Z}$.
- The reduction of A modulo $m'' := \prod_{p|\gcd(a,m)} p^{\nu_p(m)}$ has exactly one periodic point $\mathfrak{f}''$ in $\mathbb{Z}/m''\mathbb{Z}$, which we know explicitly thanks to Lemma 2.1.14.

Now, the restriction of the projection

$$\mathbb{Z}/m\mathbb{Z} \cong \mathbb{Z}/m'\mathbb{Z} \times \mathbb{Z}/m''\mathbb{Z} \to \mathbb{Z}/m'\mathbb{Z}$$

to $\mathbb{Z}/m'\mathbb{Z} \times \{\dagger''\}$ is bijective; we denote by Λ its inverse function $\mathbb{Z}/m'\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$. Then, if $\mathcal{L}'$ is a CRL-list of the reduction of A modulo m', the set $\{(\Lambda(r), l) : (r, l) \in \mathcal{L}'\}$ is a CRL-list of A. This reduces the problem to the special case where A is an affine *permutation* of a finite cyclic group, which we henceforth assume.

In order to understand CRL-lists of affine permutations of finite cyclic groups, it is helpful to proceed in several steps.

(1) First, we determine a CRL-list for each *group automorphism* of each finite *primary* cyclic group (i.e., $\mathbb{Z}/p^v\mathbb{Z}$).

(2) Next, we extend this to arbitrary affine permutations of finite *primary* cyclic groups.

(3) Finally, we use the Chinese remainder theorem and some extra ideas to construct a CRL-list of any affine permutation A of each finite cyclic group from CRL-lists of the reductions of A modulo the various prime powers $p^{v_p(m)}$.

Before tackling step (1) properly, we prove the following useful lemma.

Lemma 2.3.1. *Let X be a finite set, $\psi \in \mathrm{Sym}(X)$, $\mathcal{L}$ a CRL-list of ψ, and $n \in \mathbb{Z}$. Then the following is a CRL-list of ψ^n:*

$$\left\{\left(\psi^j(r), \frac{l}{\gcd(n,l)}\right) : (r,l) \in \mathcal{L},\ j = 0, 1, \ldots, \gcd(n,l) - 1\right\}.$$

In particular, if $\gcd(n, \mathrm{ord}(\psi)) = 1$, then $\mathcal{L}$ is also a CRL-list of ψ^n.

Proof. Each cycle of ψ^n is contained in a cycle of ψ, and for any given cycle ζ of ψ of length l, the ψ^n-cycles into which ζ decomposes correspond to the cosets of the subgroup of $\mathbb{Z}/l\mathbb{Z}$ generated by $n + l\mathbb{Z}$. Since the additive order of n modulo l is $l/\gcd(n,l)$, it follows that ζ decomposes into $\gcd(n,l)$ cycles of ψ^n, each of length $l/\gcd(n,l)$. If r is the representative of ζ from $\mathcal{L}$, then for each ψ^n-cycle ζ' contained in ζ, the elements on ζ' are just those of the form $\psi^t(r)$ for $t \in k + \gcd(n,l)\mathbb{Z}$, for some $k = k(\zeta') \in \mathbb{Z}$. It follows that the $\gcd(n,l)$ elements $\psi^j(r)$ of (the support set of) ζ for $j = 0, 1, \ldots, \gcd(n,l) - 1$ lie on pairwise distinct ψ^n-cycles and thus form a system of representatives for the ψ^n-cycles contained in ζ. This proves the main statement of the lemma. The "In particular" statement follows because the equality $\gcd(n, \mathrm{ord}(\psi)) = 1$ is equivalent to "$\gcd(n,l) = 1$ for each cycle length l of ψ". ∎

We are now ready to specify a CRL-list for each automorphism of each finite primary cyclic group. In the proof of the following lemma and beyond, we use the notation $H \leq G$ for "H is a subgroup of G", and $\langle g_1, g_2, \ldots, g_n \rangle$ to denote the subgroup of the group G generated by the elements $g_1, g_2, \ldots, g_n \in G$.

Case for p and a	elements of $\mathcal{L}(p^v, a)$
$p > 2$	$\left(\mathrm{r}^j\, p^t, \frac{\phi(p^{v-t})}{\gcd\left(\frac{\phi(p^v)}{\mathrm{ord}(a)}, \phi(p^{v-t})\right)}\right)$ for $t = 0, 1, \ldots, v$ and $j = 0, 1, \ldots, \gcd\left(\frac{\phi(p^v)}{\mathrm{ord}(a)}, \phi(p^{v-t})\right) - 1$.
$p = 2, a \equiv 1 \pmod 4$	$(0, 1)$, $(2^{v-1}, 1)$; $\left(5^j 2^t, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{\mathrm{ord}(a)}, 2^{v-t-2}\right)}\right)$, $\left(-5^j 2^t, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{\mathrm{ord}(a)}, 2^{v-t-2}\right)}\right)$ for $t = 0, 1, \ldots, v-2$ and $j = 0, 1, \ldots, \gcd\left(\frac{2^{v-2}}{\mathrm{ord}(a)}, 2^{v-t-2}\right) - 1$.
$p = 2, a \equiv 3 \pmod 4$	$(0, 1)$, $(2^{v-1}, 1)$; $(j \cdot \mathrm{ord}(-a), 2)$ for $j = 1, 2, \ldots, \frac{2^{v-1}}{\mathrm{ord}(-a)} - 1$; $\left(5^j 2^t, \frac{\mathrm{ord}(-a)}{2^t}\right)$, $\left(-5^j 2^t, \frac{\mathrm{ord}(-a)}{2^t}\right)$ for $t = 0, 1, \ldots, \log_2(\mathrm{ord}(-a)) - 1$ and $j = 0, 1, \ldots, \frac{2^{v-2}}{\mathrm{ord}(-a)} - 1$.

Table 2.1. CRL-lists of automorphisms of finite primary cyclic groups.

Lemma 2.3.2. *Let p be a prime, $v \in \mathbb{N}^+$, and $a \in \mathbb{Z}$ with $p \nmid a$. If p is odd, let r be a fixed primitive root modulo p^v, and let ϕ denote Euler's totient function. Table 2.1 provides a CRL-list $\mathcal{L}(p^v, a)$ of the automorphism $\mu_a : x \mapsto ax$ of $\mathbb{Z}/p^v\mathbb{Z}$. We write* $\mathrm{ord}(a) = \mathrm{ord}(\mu_a)$ *for the* multiplicative *order of a modulo p^v.*

Proof of Lemma 2.3.2. First, we assume that $p > 2$. If a is a primitive root modulo p^v, then the cyclic group $\langle a \rangle = (\mathbb{Z}/p^v\mathbb{Z})^* \cong \mathrm{Aut}(\mathbb{Z}/p^v\mathbb{Z})$ acts transitively on each subset of $\mathbb{Z}/p^v\mathbb{Z}$ consisting of all elements of a given additive order. Indeed, on the one hand, automorphisms of $\mathbb{Z}/p^v\mathbb{Z}$ must preserve the additive order of elements, and conversely, if $x, y \in \mathbb{Z}/p^v\mathbb{Z}$ are of the same order, then they are multiples of each other. Hence $y = z \cdot x$ for some $z \in (\mathbb{Z}/p^v\mathbb{Z})^*$. Since z is a power of a, the transitivity assertion follows. We conclude that if a is a primitive root modulo p^v, then $\mathcal{L}(p^v, a)$ may be chosen as $\{(p^t, \phi(p^{v-t})) : t = 0, 1, \ldots, v\}$, which matches with Table 2.1.

For general a, we note that a and $\mathrm{r}^{\phi(p^v)/\,\mathrm{ord}(a)}$ are powers of each other, whence by the "In particular" of Lemma 2.3.1 we may assume without loss of generality that $a = \mathrm{r}^{\phi(p^v)/\,\mathrm{ord}(a)}$. The claim now follows by applying the main statement of Lemma 2.3.1 with $n := \phi(p^v)/\,\mathrm{ord}(a)$ and $\mathcal{L} := \{(p^t, \phi(p^{v-t})) : t = 0, 1, \ldots, v\}$.

Now we assume that $p = 2$. First, let us discuss the case $a = 5$. The automorphism $\mu_5 : x \mapsto 5x$, like any automorphism of $\mathbb{Z}/2^v\mathbb{Z}$, fixes the unique elements 0 and 2^{v-1} of additive orders 1 and 2, respectively. It also fixes the order 4 elements 2^{v-2} and $3 \cdot 2^{v-2} = -2^{v-2}$. Moreover, we claim that for each $t' \in \{3, 4, \ldots, v\}$, the automorphism μ_5 has exactly two cycles on the elements of $\mathbb{Z}/2^v\mathbb{Z}$ of additive

order $2^{t'}$, both of length $2^{t'-2}$ and spanned by $2^{v-t'}$ and $-2^{v-t'}$, respectively. Indeed, this is clear for $t' = 3$ and $t' = v$; for the latter, we use that the multiplicative order of $5 = 1 + 2^2$ modulo 2^v is 2^{v-2}, that $\langle 5 \rangle \leq (\mathbb{Z}/2^v\mathbb{Z})^*$ acts semiregularly (i.e., such that no element of that group except the neutral element 1 admits fixed points in that action) on the set of generators (i.e., elements of additive order 2^v) of $\mathbb{Z}/2^v\mathbb{Z}$, and that $1 \not\equiv -1 \pmod 4$. For each other value of t', denoting by $\operatorname{aord}(x)$ the additive order of x modulo 2^v, it follows from the commutativity of the diagram

$$\begin{array}{ccc} \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 2^v\} & \xrightarrow{x \mapsto 5x} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 2^v\} \\ \Big\downarrow{\scriptstyle x \mapsto 2^{v-t'}x} & & \Big\downarrow{\scriptstyle x \mapsto 2^{v-t'}x} \\ \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 2^{t'}\} & \xrightarrow[x \mapsto 5x]{} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 2^{t'}\} \end{array}$$

that $x \mapsto 5x$ has *at most* two cycles on the set of order $2^{t'}$ elements, namely the ones spanned by $2^{v-t'} \cdot 1 = 2^{v-t'}$ and $2^{v-t'} \cdot (-1) = -2^{v-t'}$. Likewise, the commutativity of the diagram

$$\begin{array}{ccc} \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 2^{t'}\} & \xrightarrow{x \mapsto 5x} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 2^{t'}\} \\ \Big\downarrow{\scriptstyle x \mapsto 2^{t'-3}x} & & \Big\downarrow{\scriptstyle x \mapsto 2^{t'-3}x} \\ \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 8\} & \xrightarrow[x \mapsto 5x]{} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \operatorname{aord}(x) = 8\} \end{array}$$

implies that $2^{v-t'}$ and $-2^{v-t'}$ lie on *distinct* cycles of $x \mapsto 5x$. This shows that $\mathcal{L}(2^v, 5)$ can be chosen as indicated in Table 2.1.

As for other values of a, if $a \equiv 1 \pmod 4$, then a is congruent to a power of 5 modulo 2^v, and the choice for $\mathcal{L}(2^v, a)$ specified in Table 2.1 can be derived from the one for $\mathcal{L}(2^v, 5)$ using Lemma 2.3.1 (analogously to the end of the argument for $p > 2$ above).

It remains to deal with the case $a \equiv 3 \pmod 4$. Then $-a \equiv 1 \pmod 4$. We view the automorphism μ_a of $\mathbb{Z}/2^v\mathbb{Z}$ as the composition of the automorphisms μ_{-a} and μ_{-1}. For each $t' \in \{0, 1, \ldots, v\}$, we want to understand the cycles of μ_a on the set of elements x of additive order $2^{t'}$, and we do so by distinguishing some cases for t'.

- If $t' \in \{0, 1\}$ (i.e., $x \in \{0, 2^{v-1}\}$), then x, being the only element in $\mathbb{Z}/2^v\mathbb{Z}$ of its additive order, is fixed by μ_a.
- If $t' \in \{2, 3, \ldots, v - \log_2(\operatorname{ord}(-a))\}$, then μ_{-a} fixes each element of order $2^{t'}$; this can be seen by using the formula for "$a \equiv 1 \pmod 4$" in Table 2.1 and noting

that if $x \in \{a^j 2^t, -a^j 2^t\}$, then $2^{t'} = \operatorname{aord}(x) = 2^{v-t}$. It follows that the restriction of μ_a to set of order $2^{t'}$ elements is the same as that of μ_{-1}. Therefore,

$$\left\{(j \cdot \operatorname{ord}(-a), 2) : j = 1, 2, \ldots, \frac{2^{v-1}}{\operatorname{ord}(-a)} - 1\right\},$$

which is a CRL-list of the restriction of μ_{-1} to the set of elements of $\mathbb{Z}/2^v\mathbb{Z}$ with order in $\{2^2, 2^3, \ldots, 2^{v-\log_2(\operatorname{ord}(-a))}\}$, is also a CRL-list of the corresponding restriction of μ_a.

- Finally, if $t' > v - \log_2(\operatorname{ord}(-a))$, then all cycles of μ_{-a} on the set of order $2^{t'}$ elements in $\mathbb{Z}/2^v\mathbb{Z}$ are of even length (in fact, their length is a nontrivial power of 2). For a fixed $j \in \{0, 1, \ldots, \gcd(\frac{2^{v-2}}{\operatorname{ord}(-a)}, 2^{t'-2}) - 1\} = \{0, 1, \ldots, \frac{2^{v-2}}{\operatorname{ord}(-a)} - 1\}$, we consider the two cycles of μ_{-a} spanned by $5^j 2^{v-t'}$ and $-5^j 2^{v-t'}$, respectively. These cycles are distinct (according to the case "$a \equiv 1 \pmod 4$" in Table 2.1, applied to $-a$), both have length

$$\frac{2^{t'-2}}{\gcd\left(\frac{2^{v-2}}{\operatorname{ord}(-a)}, 2^{t'-2}\right)} = \frac{2^{t'-2}}{2^{v-2-\log_2(\operatorname{ord}(-a))}} = 2^{t'-v+\log_2(\operatorname{ord}(-a))}$$

and are images of each other under μ_{-1}. In fact, since μ_{-a} and μ_{-1} commute, we have $\mu_{-1}(\mu_{-a}^n(\pm 5^j 2^{v-t'})) = \mu_{-a}^n(\mu_{-1}(\pm 5^j 2^{v-t'}))$ for each $n \in \mathbb{Z}$, which leads to the following diagrammatic picture of how the cycles are matched under μ_{-1}, setting $k := 2^{t'-v+\log_2(\operatorname{ord}(-a))-1}$, so that k is half of the common cycle length of $\pm 5^j 2^{v-t'}$ under μ_{-a}:

$$\begin{array}{ccccccccc}
5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & (-a)5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & \cdots & \xrightarrow{\mu_{-a}} & (-a)^{2k-1}5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & 5^j 2^{v-t'} \\
\downarrow{\scriptstyle \mu_{-1}} & & \downarrow{\scriptstyle \mu_{-1}} & & \downarrow{\scriptstyle \mu_{-1}} & & & & \downarrow{\scriptstyle \mu_{-1}} \\
-5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & -(-a)5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & \cdots & \xrightarrow{\mu_{-a}} & -(-a)^{2k-1}5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & -5^j 2^{v-t'}
\end{array}$$

It follows that $\mu_a = \mu_{-1} \circ \mu_{-a}$ decomposes into two cycles on the union of the support sets of the above two cycles of μ_{-a}: one is spanned by $5^j 2^{v-t'}$ and consists of the "even elements" $5^j 2^{v-t'}, (-a)^2 5^j 2^{v-t'}, (-a)^4 5^j 2^{v-t'}, \ldots$ of the upper cycle as well as the "odd elements" of the lower cycle, whereas the other is spanned by $-5^j 2^{v-t'}$ and consists of the "odd elements" of the upper and "even elements" of the lower cycle. If we let t' and j run through their respective range, the corresponding cycle pairs partition the set of all elements of additive order larger than $2^v / \operatorname{ord}(-a)$. Together with the observations for smaller values of t' from above, we obtain a CRL-list for μ_a, which can be easily checked to coincide with the one specified in Table 2.1 (we note that $t' = v - t$ in the notation of that table). ∎

Now we tackle step (2) in our plan for this section, i.e., working out a CRL-list for every affine permutation A of every finite primary cyclic group. A useful observation, which we explain in more detail, is that the case where A has a fixed point can be reduced to the automorphism case (i.e., to Lemma 2.3.2). This idea appears in [21, proof of Lemma 4], which was concerned with cyclic groups, but it can easily be extended to general groups. We remind the reader that we write ρ_{r} for the right-regular representation of a group (that must be clear from context) on itself.

Lemma 2.3.3. *Let G be a group, $b \in G$, α an automorphism of G, A the affine permutation $x \mapsto x^{\alpha}b$ of G, and $\mathfrak{f} \in G$ a fixed point of A. Then*

$$A = \rho_{\mathrm{r}}(\mathfrak{f})^{-1}\alpha\rho_{\mathrm{r}}(\mathfrak{f}).$$

In particular, $\rho_{\mathrm{r}}(\mathfrak{f})$ is a digraph isomorphism from Γ_{α} to Γ_A.

Proof. The equality $\mathfrak{f}^A = \mathfrak{f}$ is equivalent to $b = (\mathfrak{f}^{-1})^{\alpha}\mathfrak{f}$. For each $x \in G$, we have

$$x^{\rho_{\mathrm{r}}(\mathfrak{f})^{-1}\alpha\rho_{\mathrm{r}}(\mathfrak{f})} = (x\mathfrak{f}^{-1})^{\alpha\rho_{\mathrm{r}}(\mathfrak{f})} = x^{\alpha}(\mathfrak{f}^{-1})^{\alpha}\mathfrak{f} = x^{\alpha}b = x^A,$$

as required. The "In particular" statement follows because of the well-known (and easy to verify) fact that for each set X and all $\psi, \psi' \in \mathrm{Sym}(X)$, the permutation ψ maps $x \in X$ to $y \in X$ if and only if its ψ'-conjugate $(\psi')^{-1}\psi\psi'$ maps $x^{\psi'}$ to $y^{\psi'}$. ∎

Lemma 2.3.3 is interesting for us because of the following elementary observation.

Lemma 2.3.4. *Let X be a finite set, and let $\psi_1, \psi_2 \in \mathrm{Sym}(X)$ be conjugate permutations, say*

$$\psi_2 = (\psi')^{-1}\psi_1\psi'.$$

If $\mathcal{L}$ is a CRL-list for ψ_1, then $\{(r^{\psi'}, l) : (r, l) \in \mathcal{L}\}$ is a CRL-list for ψ_2.

Proof. This is clear because $(y_0, y_1, \ldots, y_{l-1})$ is a cycle of ψ_1 if and only if

$$(y_0^{\psi'}, y_1^{\psi'}, \ldots, y_{l-1}^{\psi'})$$

is a cycle of ψ_2 (see the last sentence in the proof of Lemma 2.3.3). ∎

Through combining Lemmas 2.3.3 and 2.3.4, we get the next result.

Lemma 2.3.5. *Let G be a finite group, $b \in G$, α an automorphism of G, A the affine permutation $x \mapsto x^{\alpha}b$ of G, and $\mathfrak{f} \in G$ a fixed point of A. If $\mathcal{L}$ is a CRL-list of α, then $\{(r\mathfrak{f}, l) : (r, l) \in \mathcal{L}\}$ is a CRL-list of A.*

We are now ready to construct a CRL-list for each affine permutation of each finite primary cyclic group.

No.	Case for p^v, a, b	elements of $\mathcal{L}(p^v, a, b)$
1	$p > 2$, $\nu_p^{(v)}(b) \geq \nu_p^{(v)}(a-1)$	$\Big(\mathfrak{r}^j\, p^t + \mathfrak{f}, \frac{\phi(p^{v-t})}{\gcd\left(\frac{\phi(p^v)}{\mathrm{ord}(a)}, \phi(p^{v-t})\right)}\Big)$ for $t = 0, 1, \ldots, v$ and $j = 0, 1, \ldots, \gcd\left(\frac{\phi(p^v)}{\mathrm{ord}(a)}, \phi(p^{v-t})\right) - 1$.
2	$p > 2$, $\nu_p^{(v)}(b) < \nu_p^{(v)}(a-1)$	$(j, p^{v-\nu_p^{(v)}(b)})$ for $j = 0, 1, \ldots, p^{\nu_p^{(v)}(b)} - 1$.
3	$p = 2, v \leq 2, a = 1$	$(j, \mathrm{aord}(b))$ for $j = 0, 1, \ldots, \frac{2^v}{\mathrm{aord}(b)} - 1$.
4	$p^v = 4, a = 3, b = 0$	$(0, 1), (1, 2), (2, 1)$.
5	$p^v = 4, a = 3, b = 2$	$(0, 2), (1, 1), (3, 1)$.
6	$p^v = 4, a = 3, 2 \nmid b$	$(0, 2), (2, 2)$.
7	$p = 2, v \geq 3$, $\nu_2^{(v)}(b) \geq \nu_2^{(v)}(a-1)$, $a \equiv 1 \pmod 4$	$(\mathfrak{f}, 1), (2^{v-1} + \mathfrak{f}, 1)$; $\Big(5^j 2^t + \mathfrak{f}, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{\mathrm{ord}(a)}, 2^{v-t-2}\right)}\Big)$, $\Big(-5^j 2^t + \mathfrak{f}, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{\mathrm{ord}(a)}, 2^{v-t-2}\right)}\Big)$ for $t = 0, 1, \ldots, v-2$ and $j = 0, 1, \ldots, \gcd\left(\frac{2^{v-2}}{\mathrm{ord}(a)}, 2^{v-t-2}\right) - 1$.
8	$p = 2, v \geq 3$, $\nu_2^{(v)}(b) \geq \nu_2^{(v)}(a-1)$, $a \equiv 3 \pmod 4$	$(\mathfrak{f}, 1), (2^{v-1} + \mathfrak{f}, 1)$; $(j \cdot \mathrm{ord}(-a) + \mathfrak{f}, 2)$ for $j = 1, 2, \ldots, \frac{2^{v-1}}{\mathrm{ord}(-a)} - 1$; $\big(5^j 2^t + \mathfrak{f}, \frac{\mathrm{ord}(-a)}{2^t}\big), \big(-5^j 2^t + \mathfrak{f}, \frac{\mathrm{ord}(-a)}{2^t}\big)$ for $t = 0, 1, \ldots, \log_2(\mathrm{ord}(-a)) - 1$ and $j = 0, 1, \ldots, \frac{2^{v-2}}{\mathrm{ord}(-a)} - 1$.
9	$p = 2, v \geq 3$, $\nu_2^{(v)}(b) < \nu_2^{(v)}(a-1)$, $a \equiv 1 \pmod 4$	$(j, 2^{v-\nu_2^{(v)}(b)})$ for $j = 0, 1, \ldots, 2^{\nu_2^{(v)}(b)} - 1$.
10	$p = 2, v \geq 3$, $\nu_2^{(v)}(b) < \nu_2^{(v)}(a-1)$, $a \equiv 3 \pmod 4$	$(b \cdot j, 2\,\mathrm{ord}(-a))$ for $j = 1, 2, 3, 4, \ldots, \frac{2^{v-1}}{\mathrm{ord}(-a)}$.

Table 2.2. CRL-lists of affine permutations of finite primary cyclic groups.

Proposition 2.3.6. *Let p be a prime, $v \in \mathbb{N}^+$, and $a, b \in \mathbb{Z}$ with $p \nmid a$. Table 2.2 provides a CRL-list $\mathcal{L}(p^v, a, b)$ for the affine permutation $A : x \mapsto ax + b$ of $\mathbb{Z}/p^v\mathbb{Z}$, using the following conditional notations.*

- *If p is odd, we denote by $\mathfrak{r}$ a fixed primitive root modulo p^v.*
- *If $\nu_p^{(v)}(b) \geq \nu_p^{(v)}(a-1)$, we set*

$$\mathfrak{f} := -\frac{b}{p^{\nu_p^{(v)}(a-1)}} \cdot \mathrm{inv}_{p^{v-\nu_p^{(v)}(a-1)}}\left(\frac{a-1}{p^{\nu_p^{(v)}(a-1)}}\right).$$

Proof. First, we observe that if $\nu_p^{(v)}(b) \geq \nu_p^{(v)}(a-1)$, then † is a fixed point of A. Indeed, $x \in \mathbb{Z}/p^v\mathbb{Z}$ is a fixed point of A if and only if

$$\begin{aligned} &ax + b \equiv x \pmod{p^v} \Leftrightarrow (a-1)x \equiv -b \pmod{p^v} \\ &\Leftrightarrow \frac{a-1}{p^{\nu_p^{(v)}(a-1)}} x \equiv -\frac{b}{p^{\nu_p^{(v)}(a-1)}} \pmod{p^{v-\nu_p^{(v)}(a-1)}} \\ &\Leftrightarrow x \equiv -\frac{b}{p^{\nu_p^{(v)}(a-1)}} \cdot \operatorname{inv}_{p^{v-\nu_p^{(v)}(a-1)}}\left(\frac{a-1}{p^{\nu_p^{(v)}(a-1)}}\right) = \dagger \pmod{p^{v-\nu_p^{(v)}(a-1)}}. \end{aligned}$$

The form of the CRL-list for A specified in cases 1, 7 and 8 in Table 2.2 thus follows from Lemma 2.3.5 and the corresponding CRL-list for μ_a, read off from Table 2.1. Moreover, cases 3–6 in Table 2.2 are easy to check separately. It remains to justify the specified CRL-list in cases 2, 9 and 10 in Table 2.2, which we do now.

- Case 2: We note that in this case, $a \equiv 1 \pmod{p}$ necessarily. The units modulo p^v that are congruent to 1 modulo p form the unique, cyclic Sylow p-subgroup of $(\mathbb{Z}/p^v\mathbb{Z})^*$, of order p^{v-1}. For each $t \in \{0, 1, \ldots, v-1\}$, the unit $1 + p^{v-t}$ has order p^t, and thus all order p^t units modulo p^v are powers of $1 + p^{v-t}$ with exponent coprime to p and vice versa. Therefore, using the "In particular" statement of Lemma 2.3.1 and that $\operatorname{ord}(A)$ is a power of p, we may assume without loss of generality that $a = 1 + p^{v-t}$ for some $t \in \{0, 1 \ldots, v-1\}$. We observe that $v - t = \nu_p^{(v)}(a-1)$, and thus $v - t > \nu_p^{(v)}(b)$ by the case assumptions. For each $x \in \mathbb{Z}/p^v\mathbb{Z}$, we have $A(x) = ax + b = (1 + p^{v-t})x + b \equiv x \pmod{p^{\nu_p^{(v)}(b)}}$. Hence, the elements $0, 1, \ldots, p^{\nu_p^{(v)}(b)} - 1$ lie on pairwise distinct cycles of A. On the other hand, by [15, Table 3], A has exactly $p^v / \operatorname{aord}(b) = p^{\nu_p^{(v)}(b)}$ cycles, each of length $\operatorname{aord}(b) = p^{v-\nu_p^{(v)}(b)}$, so the said elements are representatives for all cycles of A and $\{(j, p^{v-\nu_p^{(v)}(b)}) : j = 0, 1, \ldots, p^{\nu_p^{(v)}(b)} - 1\}$ is a CRL-list of A, as required.
- Case 9: This can be dealt with similarly to case 2. We observe that the unit $5 = 1 + 2^2$ has multiplicative order 2^{v-2} and generates an index 2 cyclic subgroup of $(\mathbb{Z}/2^v\mathbb{Z})^*$, which consists precisely of those units that are congruent to 1 modulo 4. For each $t \in \{0, 1, \ldots, v-2\}$, the unit $1 + 2^{v-t}$ lies in this subgroup and has order 2^t, so any unit of order 2^t that is congruent to 1 modulo 4 is a power of $1 + 2^{v-t}$ with odd exponent and vice versa. Using the "In particular" statement of Lemma 2.3.1 and that $\operatorname{ord}(A)$ is a power of 2, we may assume without loss of generality that $a = 1 + 2^{v-t}$ for some $t \in \{0, 1, \ldots, v-2\}$, and the remainder of this argument is analogous to the one for case 2, resulting in $\{(j, 2^{v-\nu_2^{(v)}(b)}) : j = 0, 1, \ldots, 2^{\nu_2^{(v)}(b)} - 1\}$ being a CRL-list of A.
- Case 10: Due to $-a \equiv 1 \pmod{4}$, we may assume without loss of generality that $-a = 1 + 2^{v-t}$ for some $t \in \{0, 1, \ldots, v-2\}$ (see the argument for case 9). Let A'

be the affine function $x \mapsto -x + b$ of $\mathbb{Z}/2^{v-t}\mathbb{Z}$. For each $x \in \mathbb{Z}/2^{v}\mathbb{Z}$, we have $A(x) = -(1 + 2^{v-t})x + b \equiv -x + b \pmod{2^{v-t}}$. This means that elements of $\mathbb{Z}/2^{v-t}\mathbb{Z}$ that lie on different cycles of A' also lie on different cycles of A (we remind the reader that $\mathbb{Z}/2^{v-t}\mathbb{Z} \subseteq \mathbb{Z}/2^{v}\mathbb{Z}$ by our convention on the underlying set of $\mathbb{Z}/m\mathbb{Z}$ stated at the beginning of this section). Now, A' is an involution without fixed points (because $2 \nmid b$) and thus consists of 2^{v-t-1} transpositions. But by [15, Table 4], A has exactly 2^{v-t-1} cycles. Indeed, if $a = -5^n$, then $2^t = \operatorname{ord}(-a) = 2^{v-2-\nu_2^{(v-2)}(n)}$, and therefore $t = v - 2 - \nu_2^{(v-2)}(n)$, whence the cycle number $2^{1+\nu_2^{(v-2)}(n)}$ specified in [15, Table 4] equals 2^{v-t-1}. Therefore, any set of representatives for the cycles of A' on $\mathbb{Z}/2^{v-t}\mathbb{Z}$ is also a set of representatives for the cycles of A on $\mathbb{Z}/2^{v}\mathbb{Z}$, all of which are of length 2^{t+1} by [15, Table 4]. Thus, in order to find a CRL-list for A, it suffices to find cycle representatives for A'. To that end, we first assume that $b = 1$. Then every cycle (i.e., transposition) of A' on $\mathbb{Z}/2^{v-t}\mathbb{Z}$ contains exactly one element from the "left half" $\{1, 2, \ldots, 2^{v-t-1}\}$ and one from the "right half" $\{2^{v-t-1} + 1, 2^{v-t-1} + 2, \ldots, 2^{v-t} - 1, 2^{v-t} \equiv 0\}$, and the elements 0 and 1 lie on the same cycle. It follows that $\{1, 2, 3, 4, \ldots, 2^{v-t-1}\}$ is a set of representatives for the cycles of A', and this matches with the CRL-list for A specified in Table 2.2. For general b, we observe that

$$A' = (x \mapsto -x + b) = (x \mapsto b^{-1}x) \cdot (x \mapsto -x + 1) \cdot (x \mapsto bx),$$

whence Lemma 2.3.4 allows us to conclude that

$$b \cdot \{1, 2, 3, 4, \ldots, 2^{v-t-1}\} = \{b, 2b, 3b, 4b, \ldots, 2^{v-t-1}b\}$$

is a set of representatives for the cycles of A', as required. ∎

Now that we know a CRL-list for each affine permutation of each finite *primary* cyclic group, let us discuss how to deal with general finite cyclic groups. Through identifying the group $\mathbb{Z}/m\mathbb{Z}$ with the direct product $\prod_{p|m} \mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$ via the Chinese remainder theorem, we can view any affine permutation $A : x \mapsto ax + b$ of $\mathbb{Z}/m\mathbb{Z}$ as the "function tensor product" $\bigotimes_{p|m} A_p$, where A_p is the *reduction of A modulo $p^{\nu_p(m)}$*, i.e., the affine permutation $x \mapsto ax + b$ of $\mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$, as introduced in Remark 2.1.2. That is, A becomes the component-wise application of its reductions A_p under this identification. This leads to the following, more general problem, which we solve next.

Problem 2.3.7. *Given finite sets $X_1, X_2, \ldots, X_n$, permutations $\psi_j \in \operatorname{Sym}(X_j)$ for $j = 1, 2, \ldots, n$, and a CRL-list $\mathfrak{L}_j$ of ψ_j for $j = 1, 2, \ldots, n$, construct a CRL-list $\mathfrak{L}$ of $\psi := \bigotimes_{j=1}^{n} \psi_j \in \operatorname{Sym}(\prod_{j=1}^{n} X_j)$.*

For the rest of this section, we use the notation fixed in Problem 2.3.7. We denote by $\mathfrak{L}_j^{(1)} \subseteq X_j$ the set of first entries of the pairs in $\mathfrak{L}_j$ (i.e., the set of cycle representatives of ψ_j exhibited by $\mathfrak{L}_j$), and for $r \in \mathfrak{L}_j^{(1)}$, we denote by $r^{\langle \psi_j \rangle}$ the orbit of r

under the action of the permutation group $\langle \psi_j \rangle$ (i.e., the set of points on the ψ_j-cycle of r).

For each $\vec{r} = (r_1, r_2, \ldots, r_n) \in \prod_{j=1}^n \mathcal{L}_j^{(1)}$, we set $B_{\vec{r}} := \prod_{j=1}^n r_j^{\langle \psi_j \rangle}$. These sets $B_{\vec{r}}$ form a partition of $\prod_{j=1}^n X_j$, and each set $B_{\vec{r}}$ is a union of cycles of ψ. Therefore, it suffices to find a CRL-list $\mathcal{L}_{\vec{r}}$ of the restriction $\psi_{|B_{\vec{r}}}$ for each $\vec{r}$, then set $\mathcal{L} := \bigcup_{\vec{r}} \mathcal{L}_{\vec{r}}$.

Let us thus assume that $\vec{r}$ is fixed. For $j = 1, 2, \ldots, n$, we denote by $l_j = l_j(\vec{r})$ the ψ_j-cycle length of r_j. Every cycle of ψ on $B_{\vec{r}}$ has length $l_{\vec{r}} := \mathrm{lcm}(l_1, l_2, \ldots, l_n)$, and there are exactly $(\prod_{j=1}^n l_j)/l_{\vec{r}}$ such cycles (see also [86, Lemma 2.1]). It remains to find representatives for them.

We consider the bijection

$$\iota_{\vec{r}} : \prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z} \to B_{\vec{r}}, \quad (k_1, k_2, \ldots, k_n) \mapsto (\psi_1^{k_1}(r_1), \psi_2^{k_2}(r_2), \ldots, \psi_n^{k_n}(r_n)).$$

If we identify $B_{\vec{r}}$ with $\prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z}$ via this bijection, then the action of ψ on $B_{\vec{r}}$ turns into that of the function

$$\mathfrak{s}_{\vec{r}} : \prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z} \to \prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z}, \quad (k_1, k_2, \ldots, k_n) \mapsto (k_1 + 1, k_2 + 1, \ldots, k_n + 1),$$

each displayed addition being modulo the corresponding l_j, of course. So it suffices to find a set of representatives for the cycles of $\mathfrak{s}_{\vec{r}}$ on $\prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z}$, then map that set under $\iota_{\vec{r}}$. In order to describe a particular set of cycle representatives for $\mathfrak{s}_{\vec{r}}$ neatly, we introduce the following auxiliary concepts.

Definition 2.3.8. We denote by $\pi(l_{\vec{r}})$ the set of all prime divisors of $l_{\vec{r}}$.

(1) A function $\mathcal{I} : \pi(l_{\vec{r}}) \to \{1, 2, \ldots, n\}$ is an *$\vec{r}$-admissible indexing function* if for each $p \in \pi(l_{\vec{r}})$ we have $v_p(l_{\mathcal{I}(p)}) = \max\{v_p(l_j) : j = 1, 2, \ldots, n\}$.

(2) If $\mathcal{I}$ is an $\vec{r}$-admissible indexing function, then a tuple

$$(k_1, k_2, \ldots, k_n) \in \prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z}$$

is *$\mathcal{I}$-good* if $k_{\mathcal{I}(p)} \equiv 0 \pmod{p^{v_p(l_{\mathcal{I}(p)})}}$ for each $p \in \pi(l_{\vec{r}})$.

(3) For each $\vec{r}$-admissible indexing function $\mathcal{I}$, we denote by $\mathrm{Good}_{\vec{r}}(\mathcal{I})$ the set of all $\mathcal{I}$-good tuples in $\prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z}$.

The following result solves Problem 2.3.7.

Proposition 2.3.9. *Let $\mathcal{I}$ be an $\vec{r}$-admissible indexing function. Then* $\mathrm{Good}_{\vec{r}}(\mathcal{I})$ *is a set of representatives for the cycles of $\mathfrak{s}_{\vec{r}}$ on $\prod_{j=1}^n \mathbb{Z}/l_j\mathbb{Z}$. Equivalently,*

$$\iota_{\vec{r}}(\mathrm{Good}_{\vec{r}}(\mathcal{I})) \times \{l_{\vec{r}}\}$$

is a CRL-list for $\psi_{|B_{\vec{r}}}$, and so

$$\bigcup_{\vec{r}\in\prod_{j=1}^{n}\mathscr{L}_j^{(1)}} (\iota_{\vec{r}}(\mathrm{Good}_{\vec{r}}(\mathcal{I}))\times\{l_{\vec{r}}\})$$

is a CRL-list for ψ.

Proof. By definition and the Chinese remainder theorem, the number of $\mathcal{I}$-good tuples in $\prod_{j=1}^{n}\mathbb{Z}/l_j\mathbb{Z}$ is

$$\frac{\prod_{j=1}^{n} l_j}{\prod_{p\in\pi(l_{\vec{r}})} p^{v_p(l_{\mathcal{I}(p)})}} = \frac{|\prod_{j=1}^{n}\mathbb{Z}/l_j\mathbb{Z}|}{l_{\vec{r}}},$$

which is also the number of cycles of $\mathfrak{s}_{\vec{r}}$ on $\prod_{j=1}^{n}\mathbb{Z}/l_j\mathbb{Z}$. Hence, it suffices to show that different $\mathcal{I}$-good tuples lie on distinct cycles of $\mathfrak{s}_{\vec{r}}$. Let $\vec{k}=(k_1,k_2,\dots,k_n)$ and $\vec{k}'=(k_1',k_2',\dots,k_n')$ be $\mathcal{I}$-good tuples that lie on the same cycle of $\mathfrak{s}_{\vec{r}}$. This means that there is a $t\in\mathbb{Z}$ such that $k_j+t\equiv k_j' \pmod{l_j}$ for each $j=1,2,\dots,n$. Now, let $p\in\pi(l_{\vec{r}})$. Since $t\equiv k'_{\mathcal{I}(p)}-k_{\mathcal{I}(p)} \pmod{l_{\mathcal{I}(p)}}$ and $\vec{k},\vec{k}'$ are $\mathcal{I}$-good, it follows that $t\equiv 0 \pmod{p^{v_p(l_{\mathcal{I}(p)})}}$. Because this holds for every $p\in\pi(l_{\vec{r}})$, we conclude that

$$\mathrm{lcm}(l_1,l_2,\dots,l_n)=l_{\vec{r}}=\prod_{p\in\pi(l_{\vec{r}})} p^{v_p(l_{\mathcal{I}(p)})}$$

divides t, whence $k_j\equiv k_j' \pmod{l_j}$ for each $j=1,2,\dots,n$. This means that $\vec{k}=\vec{k}'$, as required. ∎

2.4 Affine discrete logarithms and cycle lengths

Let $m\geq 1$ be an integer, and let $a,b\in\mathbb{Z}/m\mathbb{Z}$ with $\gcd(a,m)=1$. We consider the affine permutation $A: x\mapsto ax+b$ of $\mathbb{Z}/m\mathbb{Z}$. Given $x,y\in\mathbb{Z}/m\mathbb{Z}$, we set

$$\log_A^{(m)}(x,y):=\begin{cases}\infty, & \text{if there is no } k\in\mathbb{Z} \text{ with } A^k(x)=y,\\ \min\{k\in\mathbb{N}_0 : A^k(x)=y\}, & \text{otherwise.}\end{cases}$$

In this short section, we discuss how to compute $\log_A^{(m)}(x,y)$ and the cycle length of x under A, which is closely related, as it is the minimal *positive* integer k such that $A^k(x)=x$ (while $\log_A^{(m)}(x,x)=0$). It is not surprising that modular discrete logarithms play an important role in this, because they are a special case of the notion $\log_A^{(m)}(x,y)$. Namely, the discrete logarithm of $x\in(\mathbb{Z}/m\mathbb{Z})^*$ modulo m with base $a\in(\mathbb{Z}/m\mathbb{Z})^*$, written $\log_a^{(m)}(x)$, is equal to $\log_{\mu_a}^{(m)}(1,x)$.

In order to discuss the computational details, we make a case distinction.

- First, we assume that $a \equiv 1 \pmod{m}$, a simple case for which no discrete logarithms need to be computed. Indeed, one then has $A^k(x) = x + kb \equiv y \pmod{m}$ if and only if $kb \equiv y - x \pmod{m}$. That last congruence is solvable in k if and only if $\gcd(b, m) \mid y - x$, in which case the congruence is equivalent to

$$\frac{b}{\gcd(b,m)}k \equiv \frac{y-x}{\gcd(b,m)} \left(\mathrm{mod}\ \frac{m}{\gcd(b,m)}\right) \tag{2.7}$$

and has the minimal solution

$$\left(\frac{y-x}{\gcd(b,m)} \cdot \mathrm{inv}_{m/\gcd(b,m)}\left(\frac{b}{\gcd(b,m)}\right)\right) \bmod \frac{m}{\gcd(b,m)} = \log_A^{(m)}(x,y).$$

We note that in case $x = y$, the minimal *positive* solution of congruence (2.7), and thus the cycle length of x under A modulo m, is $m/\gcd(b, m)$.

- Now we assume that $a \not\equiv 1 \pmod{m}$. Then $m > 1$, and $a \not\equiv 0 \pmod{m}$ due to $\gcd(a, m) = 1$, so we may assume that as an integer, $a > 1$. We have $A^k(x) = y$ if and only if

$$\begin{aligned} & a^k x + \frac{a^k - 1}{a-1} b \equiv y \pmod{m} \\ & \quad \Leftrightarrow a^k(a-1)x + (a^k - 1)b \equiv (a-1)y \pmod{(a-1)m} \\ & \quad \Leftrightarrow a^k((a-1)x + b) \equiv (a-1)y + b \pmod{(a-1)m}. \end{aligned} \tag{2.8}$$

In order for congruence (2.8) to be solvable in k, it is necessary that $(a-1)x + b$ and $(a-1)y + b$ have the same additive order modulo $(a-1)m$, i.e., that

$$\gcd((a-1)x + b, (a-1)m) = \gcd((a-1)y + b, (a-1)m) =: \mathfrak{d}. \tag{2.9}$$

If condition (2.9) is satisfied, then congruence (2.8) is equivalent to

$$a^k \cdot \frac{(a-1)x + b}{\mathfrak{d}} \equiv \frac{(a-1)y + b}{\mathfrak{d}} \left(\mathrm{mod}\ \frac{(a-1)m}{\mathfrak{d}}\right),$$

i.e., to

$$a^k \equiv \frac{(a-1)y + b}{\mathfrak{d}} \cdot \mathrm{inv}_{(a-1)m/\mathfrak{d}}\left(\frac{(a-1)x + b}{\mathfrak{d}}\right) \left(\mathrm{mod}\ \frac{(a-1)m}{\mathfrak{d}}\right), \tag{2.10}$$

which shows that

$$\log_A^{(m)}(x,y) = \log_a^{((a-1)m/\mathfrak{d})}\left(\frac{(a-1)y + b}{\mathfrak{d}} \cdot \mathrm{inv}_{(a-1)m/\mathfrak{d}}\left(\frac{(a-1)x + b}{\mathfrak{d}}\right)\right),$$

with the convention that $\log_a^{(m)}(x) = \infty$ if x is not a power of a modulo m. If $x = y$, then the right-hand side in congruence (2.10) simplifies to 1, whence the cycle length of x under A equals the multiplicative order of a modulo $(a-1)m/\mathfrak{d}$.

The upshot of this discussion is that $\log_A^{(m)}(x, y)$ and the cycle length of x under A modulo m can be computed efficiently if one has efficient algorithms for computing discrete logarithms and multiplicative orders of units in $(\mathbb{Z}/m\mathbb{Z})^*$. Hence, $\log_A^{(m)}(x,y)$ can be computed efficiently on a quantum computer. Indeed, Shor showed that such computers admit efficient algorithms both for computing discrete logarithms and for integer factorization [68], the latter of which is sufficient to compute element orders in $(\mathbb{Z}/m\mathbb{Z})^*$ efficiently; in fact, all one needs for that is an explicit factorization of the Euler totient function value $\phi(m)$, see also the proof of Lemma 5.1.6 (2).

Chapter 3

Functional graphs of generalized cyclotomic mappings

Let f be a generalized cyclotomic mapping of $\mathbb{F}_q$ of index d. From the introduction, we recall our notation C_i for $i \in \{0, 1, \ldots, d\}$, where $C_d = \{0_{\mathbb{F}_q}\}$, and $C_i = \omega^i C$ for $i < d$ is a coset of the index d subgroup C of $\mathbb{F}_q^* = \langle \omega \rangle$. Moreover, we recall that for each $i \in \{0, 1, \ldots, d-1\}$ we have a natural bijection $\iota_i : \mathbb{Z}/s\mathbb{Z} \to C_i = \omega^i C$, $x \mapsto \omega^{i+dx}$, by virtue of which we view C_i as a copy of $\mathbb{Z}/s\mathbb{Z}$, where

$$s = (q-1)/d = |C|.$$

As long as f does not map C_i constantly to $C_d = \{0\}$, this allows us to view the restriction $f_{|C_i}$ as an affine function $A_i : \mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}$. Finally, we recall the induced function $\bar{f} : \{0, 1, \ldots, d\} \to \{0, 1, \ldots, d\}$ (the unique function such that $f(C_i) \subseteq C_{\bar{f}(i)}$ for each i).

Our goal in this chapter is to describe methods through which the isomorphism type of the functional graph Γ_f can be understood, following the approach outlined in the introduction.

3.1 Periodic points and CRL-lists

Understanding the periodic points and finding a CRL-list of f can be reduced to the corresponding tasks for affine maps of finite cyclic groups. We observe that periodic points of f are necessarily contained in "periodic blocks" (i.e., blocks C_i such that i is periodic under $\bar{f}$). We assume that it is an easy task (due to d being sufficiently small) to find a CRL-list $\bar{\mathcal{L}}$ for $\bar{f}$. We determine the periodic points of f according to the "block cycle" of $\bar{f}$ they lie on, so let $(i, \ell) \in \bar{\mathcal{L}}$.

If $i = d$, then $\ell = 1$, and the only point to consider is the field element 0, which is by definition periodic of cycle length 1 under f. We note the contribution $\mathcal{L}_d := \{(0, 1)\}$ to the CRL-list $\mathcal{L}$ of f we are building.

Now we assume that $i < d$, and that the cycle of i under $\bar{f}$ is $(i_0, i_1, \ldots, i_{\ell-1})$ with $i_0 = i$. A point $x \in C_i$ is periodic under f if and only if it is periodic under the iterate f^ℓ, which stabilizes C_i and acts on the corresponding copy of $\mathbb{Z}/s\mathbb{Z}$ via the affine map $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$. In other words, the periodic points of f in C_i are in bijection (via ι_i) to the periodic points of $\mathcal{A}_i$ in $\mathbb{Z}/s\mathbb{Z}$, and are thus characterized by Lemma 2.1.14. We note that the set of periodic points of f in a different coset C_{i_t} of the same $\bar{f}$-cycle is simply the iterated set image $f^t(\mathrm{per}(f_{|C_i}))$. Moreover, the cycle length of a periodic point $x \in C_i$ under f is the ℓ-fold of its cycle length under $\mathcal{A}_i$. It follows that if $\mathcal{L}_i' \subseteq \mathbb{Z}/s\mathbb{Z} \times \mathbb{N}^+$ is a CRL-list of $\mathcal{A}_i$ (which we can determine as

described in Section 2.3), then $\mathcal{L}_i := \{(\iota_i(r), \ell \cdot l) : (r, l) \in \mathcal{L}'_i\}$ is CRL-list of the restriction of f to the entire "coset cycle spanned by C_i" (i.e., to the set $\bigcup_{t=0}^{\ell-1} C_{i_t}$). In summary, we obtain the following proposition.

Proposition 3.1.1. *Let $\bar{\mathcal{L}}$ be a CRL-list for $\bar{f}$. We set $\mathcal{L}_d := \{(0_{\mathbb{F}_q}, 1)\}$. Moreover, for $i < d$ with $(i, \ell) \in \bar{\mathcal{L}}$, we define $\mathcal{L}_i$ as follows. Let $(i_0, i_1, \ldots, i_{\ell-1})$ with $i_0 = i$ be the $\bar{f}$-cycle of i, and let $\mathcal{L}'_i$ be a CRL-list of the affine map $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$ of $\mathbb{Z}/s\mathbb{Z}$. Then we set $\mathcal{L}_i := \{(\iota_i(r), \ell \cdot l) : (r, l) \in \mathcal{L}'_i\}$. With this definition of $\mathcal{L}_i$ for each $(i, \ell) \in \bar{\mathcal{L}}$, we have that*

$$\mathcal{L} := \bigcup_{(i,\ell)\in\bar{\mathcal{L}}} \mathcal{L}_i$$

is a CRL-list of f.

3.2 The induced subgraph on the periodic cosets

Our next goal is to understand the trees $\mathrm{Tree}_{\Gamma_f}(x)$ in Γ_f above periodic points x of f, in the sense of Definition 1.4. We remind the reader that $\mathrm{Tree}_{\Gamma_f}(x)$ is defined for arbitrary vertices x of Γ_f, not just periodic ones. In general, it is advantageous to take a recursive approach, understanding $\mathrm{Tree}_{\Gamma_f}(x)$ for vertices x according to their depth in Γ_f, starting with leaves and working toward periodic vertices, which are at the end of the recursion. Before we carry this out, however, we must understand the induced subgraph Γ_{per} of Γ_f on the union of all periodic blocks C_i (i.e., blocks where i is periodic under $\bar{f}$) as a stepping stone.

We observe that Γ_{per} is the functional graph of the restriction f_{per} of f to the union of all periodic blocks. Just like f has the induced function $\bar{f}$ on the index set $\{0, 1, \ldots, d\}$, the restriction f_{per} has the induced function $\overline{f_{\mathrm{per}}}$, which is the restriction of $\bar{f}$ to its set of periodic points. Hence, $\overline{f_{\mathrm{per}}}$ is a permutation of its domain of definition, a fact that is important for our argument.

Similarly to the situation described in Proposition 2.1.8, if we know, for a given periodic vertex x of $\Gamma_{\mathrm{per}} = \Gamma_{f_{\mathrm{per}}}$, that each $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(y)$, where y is a child of x in Γ^*_{per}, has rigid procreation, and we know the first $h = h(y)$ procreation numbers of each child y, where h is the height of $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(y)$, then this characterizes the isomorphism type of each $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(y)$, and thus of $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(x)$, uniquely. And indeed, while Γ^*_{per} itself need not have rigid procreation in the more general setting we are considering here, the trees we just referred to do have it. More specifically, we have the following result (in which the exclusion of $i = d$ is without loss of generality, because $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(0_{\mathbb{F}_q})$ is trivial anyway).

Theorem 3.2.1. *Let f_{per} and $\overline{f_{\mathrm{per}}}$ be as above. Moreover, let $i \in \mathrm{dom}(\overline{f_{\mathrm{per}}}) = \mathrm{per}(\bar{f})$ with $i < d$, and let $(i_0, i_1, \ldots, i_{\ell-1})$ be the cycle of $i = i_0$ under $\overline{f_{\mathrm{per}}}$. We extend the*

notation i_t to arbitrary $t \in \mathbb{Z}$ by reducing t modulo ℓ (so that, for instance, $i_\ell = i_0$). For $t = 0, 1, \ldots, \ell - 1$, say $A_{i_t} : z \mapsto \alpha_{i_t} z + \beta_{i_t}$ is the affine map of $\mathbb{Z}/s\mathbb{Z}$ that describes how f_{per} (or, equivalently, f) maps from C_{i_t} to $C_{i_{t+1}}$, and let $\varphi_{i_t} := \mu_{\alpha_{i_t}} : z \mapsto \alpha_{i_t} z$, be the associated group endomorphism of $\mathbb{Z}/s\mathbb{Z}$. Then the following holds for any positive integer k. If $x \in C_i$ has at least k successor generations in Γ^*_{per} (we note that those successor generations need not be entirely contained in C_i), then

$$\mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) = \left|\ker\left(\prod_{j=0}^{k-1} \varphi_{i_{-k+j}}\right) : \ker\left(\prod_{j=0}^{k-2} \varphi_{i_{-k+j}}\right)\right| = \frac{\gcd\left(\prod_{j=0}^{k-1} \alpha_{i_{-k+j}}, s\right)}{\gcd\left(\prod_{j=0}^{k-2} \alpha_{i_{-k+j}}, s\right)},$$

independently of x.

Proof. This theorem can be seen as a generalization of Theorem 2.1.21 (which corresponds to the case $\ell = 1$), and likewise, its proof is a generalization of that of Theorem 2.1.21. We proceed by induction on k. For $k = 1$, we observe that $C_{\overline{f_{\mathrm{per}}}^{-1}(i)} = C_{i_{-1}}$ is the unique coset which f_{per} maps to C_i. Hence

$$\mathrm{proc}_1^{(\Gamma^*_{\mathrm{per}})}(x) = \#\text{ children of } x \text{ in } \Gamma^*_{\mathrm{per}} = |\{y \in \mathbb{Z}/s\mathbb{Z} : A_{i_{-1}}(y) = x\}| = |\ker(\varphi_{i_{-1}})|,$$

which implies the statement for $k = 1$ since an empty product of group endomorphisms is by definition the identity function id.

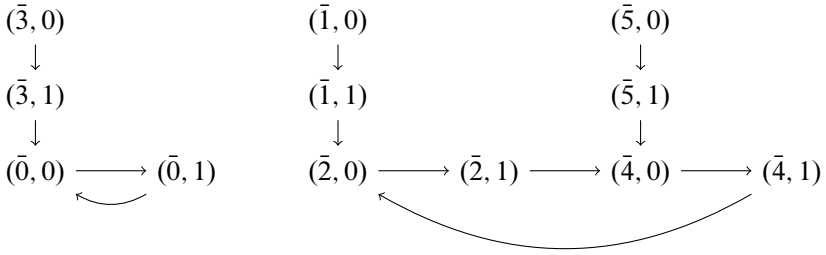
Now we assume that $k \geq 2$ and that the statement holds up to $k - 1$ for points in C_{i_t}, where $t \in \mathbb{Z}$ is arbitrary. For $h = 1, 2, \ldots, k-1$ and $t \in \mathbb{Z}$, we denote by $\mathrm{proc}_{i_t,h}$ the common procreation number $\mathrm{proc}_h^{(\Gamma^*_{\mathrm{per}})}(y)$ for all vertices $y \in C_{i_t}$ with at least h successor generations in Γ^*_{per}. For each $h = 1, 2, \ldots, k-1$, the number of endpoints of paths of length h in Γ^*_{per} starting at a vertex in C_{i_t} with at least h successor generations in Γ^*_{per} is $\prod_{j=0}^{h-1} \mathrm{proc}_{i_{t-j},h-j}$, and an easy induction on h shows that it is also equal to $|\ker(\prod_{j=0}^{h-1} \varphi_{i_{t-h+j}})|$. Using this, it follows that for each $x \in C_i = C_{i_0}$ with at least k successor generations, one has

$$\begin{aligned}
&\mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) \cdot \left|\ker\left(\prod_{j=0}^{k-2} \varphi_{i_{-k+j}}\right)\right| \\
&\quad = \mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) \cdot \prod_{j=0}^{k-2} \mathrm{proc}_{i_{-j-1},k-1-j} \\
&\quad = (\#\text{ endpoints of paths of length } k \text{ starting at } x) = \left|\ker\left(\prod_{j=0}^{k-1} \varphi_{i_{-k+j}}\right)\right|,
\end{aligned}$$

from which the asserted formula for $\mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x)$ follows readily. ∎

The following example highlights some properties that do *not* hold in general.

Example 3.2.2. Let $q = 13$, $d = 2$ (thus $s = 6$), and $\bar{f} = (0,1)(2)$, so that $\Gamma_f = \Gamma_{\text{per}}$. Moreover, we assume that $A_0(z) = z$ and $A_1(z) = 2z$. Then $f_{|\mathbb{F}_q^*}$ has the following functional graph, in which we denote the point in C_i corresponding to $z \in \mathbb{Z}/6\mathbb{Z}$ by (z, i):



We note the following.

- The rooted trees above periodic vertices in C_0 are *not* isomorphic to the rooted trees above periodic vertices in C_1.
- Transient vertices in C_1 have strictly larger tree height in Γ_f than periodic vertices in C_1. More specifically, the transient vertices in C_1 are just those with tree height 1, the periodic vertices are those with tree height 0 in Γ_f.
- The set of possible tree heights in Γ_f above vertices in C_0 is $\{0, 2\}$, which is *not* an integer interval.

3.3 The rooted trees

We describe a recursive approach for understanding $\text{Tree}_{\Gamma_f}(x)$ for each vertex $x \in \mathbb{F}_q = \text{V}(\Gamma_f)$. We proceed in three steps, according to the unique $i \in \{0, 1, \ldots, d\}$ such that $x \in C_i$. Unless $i = d$, our goal is to find an arithmetic partition $\mathcal{P}_i$ of $\mathbb{Z}/s\mathbb{Z}$, corresponding to a partition of C_i via the bijection ι_i (that we also call an *arithmetic partition of* C_i), such that for vertices $x \in C_i$ from a common block $\mathcal{B}(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ of that partition, the isomorphism type of $\text{Tree}_{\Gamma_f}(x)$ is constant, denoted by $\text{Tree}_i(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$. We also want to understand $\text{Tree}_i(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ in terms of $\vec{v}^{(\mathcal{P}_i)}$ explicitly, and verify that for fixed d, the maximum (arithmetic) complexity of $\mathcal{P}_i$ (in the sense of Definition 1.5 (3)) is in $O(d^2 \,\text{mpe}(q-1)) \subseteq O(d^2 \log q)$, where $\text{mpe}(m) := \max_p \nu_p(m)$ for $m \in \mathbb{N}^+$ (and p ranges over all primes). First, we introduce a few notations.

- For $M \subseteq \mathbb{F}_q$ and $x \in \mathbb{F}_q$, the notation $\text{Tree}_{\Gamma_f}(x, M)$ denotes the digraph isomorphism type of the subgraph of Γ_f that is a rooted tree with root x, obtained by attaching to that root all rooted trees $\text{Tree}_{\Gamma_f}(y)$, where y is an f-transient pre-image of x with $y \in M$. We note that $\text{Tree}_{\Gamma_f}(x, \mathbb{F}_q) = \text{Tree}_{\Gamma_f}(x)$.

- Let $M \subseteq \mathbb{F}_q$, and let us assume that $\mathcal{P}$ is an arithmetic partition of C_i with a fixed sequence of spanning congruences such that for $x \in C_i$, the rooted tree isomorphism type $\mathrm{Tree}_{\Gamma_f}(x, M)$ only depends on the block $\mathcal{B}(\mathcal{P}, \vec{\nu})$ of $\mathcal{P}$ in which x lies (but not on x itself). Then we denote that isomorphism type by $\mathrm{Tree}_i(\mathcal{P}, M, \vec{\nu})$. We also set $\mathrm{Tree}_i(\mathcal{P}, \vec{\nu}) := \mathrm{Tree}_i(\mathcal{P}, \mathbb{F}_q, \vec{\nu})$.
- If $\mathfrak{T}_1, \mathfrak{T}_2, \ldots, \mathfrak{T}_N$ are isomorphism types of rooted trees, then their *sum* $\mathfrak{T}_1 + \mathfrak{T}_2 + \cdots + \mathfrak{T}_N$ is defined as the rooted tree isomorphism type obtained by glueing disjoint copies of the $\mathfrak{T}_j$ together at their roots. This addition turns the class of rooted tree isomorphism types into a class-sized monoid, the neutral element of which is the trivial rooted tree isomorphism type (a single vertex without arcs).
- If $\mathfrak{T}$ is a rooted tree isomorphism type, we denote by $\mathfrak{T}^+$ the rooted tree isomorphism type obtained by connecting a copy of $\mathfrak{T}$ to a new root via an arc from the old to the new root. For example, iterating this operation starting from the trivial rooted tree isomorphism type, one obtains those finite digraphs that are directed paths.
- If $\mathfrak{T}$ is a rooted tree isomorphism type and n is a non-negative integer, we define the *multiple* $n \cdot \mathfrak{T} = n\mathfrak{T}$ as follows recursively. $0\mathfrak{T}$ is the trivial rooted tree isomorphism type, and $(n+1)\mathfrak{T} := n\mathfrak{T} + \mathfrak{T}$.
- In view of the previous two bullet points, non-negative integer linear combinations $n_1\mathfrak{T}_1 + n_2\mathfrak{T}_2 + \cdots + n_N\mathfrak{T}_N$ of rooted tree isomorphism types are well defined.
- If $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are arithmetic partitions of $\mathbb{Z}/m\mathbb{Z}$, then $\bigwedge_{k=1}^{n} \mathcal{X}_k$ denotes the infimum of the $\mathcal{X}_k$ in the lattice of all partitions of $\mathbb{Z}/m\mathbb{Z}$ (i.e., the roughest common refinement of the $\mathcal{X}_k$). Equivalently, if a spanning m-congruence sequence is fixed for each $\mathcal{X}_k$, then $\bigwedge_{k=1}^{n} \mathcal{X}_k$ is the arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ that is spanned by the concatenation of those sequences.

For the first step of our approach, we consider the case where $i \neq d$ and C_i is *not* a periodic coset (let us call such cosets *transient*). For this case, we proceed by recursion on the height of $\mathrm{Tree}_{\Gamma_{\bar{f}}}(i)$. The base of the recursion is when i is a leaf in $\Gamma_{\bar{f}}$. Then $x \in C_i$ is a leaf in Γ_f, i.e., $x \notin \mathrm{im}(f)$. This means that $\mathrm{Tree}_{\Gamma_f}(x)$ consists of the single vertex x and has no arcs. Therefore, we may choose $\mathcal{P}_i$ as the trivial partition $\mathfrak{P}(\emptyset) = \{\mathbb{Z}/s\mathbb{Z}\}$ of $\mathbb{Z}/s\mathbb{Z}$, and $\mathrm{Tree}_i(\mathcal{P}_i, \emptyset)$ as the trivial rooted tree isomorphism type. We note that $\mathrm{AC}(\mathcal{P}_i)$, the complexity of $\mathcal{P}_i$, is $0 = \mathfrak{v}_i$ where, for a general $j \in \{0, 1, \ldots, d\}$, we set $\mathfrak{v}_j := |\mathrm{V}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(j))| - 1$, the number of vertices that are strictly above j in the corresponding subtree of $\Gamma_{\bar{f}}$.

Now we assume that C_i is transient, that the height of $\mathrm{Tree}_{\Gamma_{\bar{f}}}(i)$ is $h \geq 1$, and that all transient cosets where that height is less than h have been "taken care of" via arithmetic partitions $\mathcal{P}_j$ such that $\mathrm{AC}(\mathcal{P}_j) \leq \mathfrak{v}_j$. In particular, if $\bar{f}^{-1}(\{i\}) = \{j_1, j_2, \ldots, j_K\}$, then for each $t = 1, 2, \ldots, K$, we have an arithmetic partition $\mathcal{P}_{j_t}$

of C_{j_t} with an explicit sequence of spanning s-congruences of length $m_{j_t} \leq \mathfrak{v}_{j_t}$ such that the isomorphism type $\mathrm{Tree}_{\Gamma_f}(y)$ is the same for all vertices $y \in C_{j_t}$ chosen from a common block $\mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ of $\mathcal{P}_{j_t}$, and we understand each such isomorphism type $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ explicitly. Now, because each pre-image y of $x \in C_i$ under f (i.e., each child of x in Γ_f^*) must lie in one of the cosets C_{j_t} for $t = 1, 2, \ldots, K$, it follows that

$$\mathrm{Tree}_{\Gamma_f}(x) = \sum_{t=1}^{K} \mathrm{Tree}_{\Gamma_f}(x, C_{j_t}).$$

Moreover, for fixed $t \in \{1, 2, \ldots, K\}$, we can write

$$\mathrm{Tree}_{\Gamma_f}(x, C_{j_t}) = \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} (|f^{-1}(\{x\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})| \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^+).$$

Let us consider the arithmetic partition $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$ of C_i with its explicit spanning s-congruence sequence of length $m_{j_t} + 1$ from Lemma 2.2.2. If x lies in the block $\mathcal{B}(\mathcal{P}'_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ for some fixed $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$, then

$$|f^{-1}(\{x\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})| = \sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$$

by Lemma 2.2.2, and thus

$$\mathrm{Tree}_{\Gamma_f}(x, C_{j_t}) = \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^+),$$

independently of x itself. Now, let us set $\mathcal{P}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$, viewed as an arithmetic partition of C_i with a spanning sequence of length $m_i := \sum_{t=1}^{K} (m_{j_t} + 1)$. We can view each logical sign tuple $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$ as a concatenation of logical sign tuples $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$ for $t = 1, 2, \ldots, K$, and if $x \in C_i$ lies in the block $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ of $\mathcal{P}_i$, then x also lies in the block $\mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ of $\mathcal{P}'_{j_t}$ for each $t = 1, 2, \ldots, K$, whence

$$\mathrm{Tree}_{\Gamma_f}(x) = \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^+),$$

independently of x itself, as required. Moreover, we note that

$$\mathrm{AC}(\mathcal{P}_i) \leq \sum_{t=1}^{K} \mathrm{AC}(\mathcal{P}'_{j_t}) \leq \sum_{t=1}^{K} (\mathfrak{v}_{j_t} + 1) = \mathfrak{v}_i.$$

In summary, we obtain the following result.

Proposition 3.3.1. *For each $\bar{f}$-transient $i \in \{0, 1, \ldots, d-1\}$, the arithmetic partition $\mathcal{P}_i$ of C_i together with an explicit spanning sequence of s-congruences of length m_i and associated rooted tree isomorphism types $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ for $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$ can be defined as follows by recursion on $h_i := \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i))$.*

(1) *If $h_i = 0$, we may set*
   (a) $\mathcal{P}_i := \mathfrak{P}(\emptyset)$,
   (b) $m_i := 0$, *and*
   (c) $\mathrm{Tree}_i(\mathcal{P}_i, \emptyset)$ *to be the trivial rooted tree isomorphism type.*

(2) *If $h_i \geq 1$, we let $\bar{f}^{-1}(\{i\}) = \{j_1, j_2, \ldots, j_K\}$ and set $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$ for $t = 1, 2, \ldots, K$. Then we define*
   (a) $\mathcal{P}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$,
   (b) $m_i := \sum_{t=1}^{K} (m_{j_t} + 1)$, *and*
   (c) *for $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$, viewed as the concatenation of the logical sign tuples $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$ for $t = 1, 2, \ldots, K$,*

$$\begin{aligned}&\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})\\ &:= \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset,\neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^{+}).\end{aligned}$$

With this choice of $\mathcal{P}_i$, we have

$$\mathrm{AC}(\mathcal{P}_i) \leq \mathfrak{v}_i := |\mathrm{V}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i))| - 1 \leq d - 1 \in O(d) \subseteq O(d^2 \,\mathrm{mpe}(q-1)).$$

The second step is to describe the isomorphism type of $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, which is similar to the recursion step for transient cosets above. Let

$$\bar{f}^{-1}(\{d\}) = \{d, j_1, j_2, \ldots, j_K\}$$

(we note that $K = 0$ unless some coefficient a_i in the cyclotomic form (1.1) of f is 0). The children of 0 in $\mathrm{Tree}_{\Gamma_f}(0)^*$ are just the *non-zero* children of 0 in Γ_f^*, and each such child must lie in C_{j_t} for some $t \in \{1, 2, \ldots, K\}$. We observe that each C_{j_t} is a transient coset, so by Proposition 3.3.1, we already know a suitable arithmetic partition $\mathcal{P}_{j_t}$ of C_{j_t}, with an explicit spanning sequence of length m_{j_t}, and have an explicit understanding of the rooted tree isomorphism types $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ for $\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}$. Moreover, all vertices in C_{j_t} map to $0_{\mathbb{F}_q}$ under f, so

$$f^{-1}(\{0_{\mathbb{F}_q}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})}) = \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})}) = \mathbf{0}^{-1}(\{0_{\mathbb{Z}/s\mathbb{Z}}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$$

for all $\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}$, where $\mathbf{0} : \mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}, z \mapsto 0 = 0z + 0$. Hence, if we set $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, \mathbf{0})$, which is, in its standard form from Lemma 2.2.2, spanned by

the single s-congruence $x \equiv 0 \pmod{s}$ repeated $m_{j_t} + 1$ times, then

$$|f^{-1}(\{0_{\mathbb{F}_q}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{v}^{(\mathcal{P}_{j_t})})| = \sigma_{\mathcal{P}_{j_t},\mathbf{0}}(\vec{v}^{(\mathcal{P}_{j_t})}, (\emptyset, \dots, \emptyset)).$$

Since $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ is obtained by attaching $|f^{-1}(\{0_{\mathbb{F}_q}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{v}^{(\mathcal{P}_{j_t})})|$ copies of $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{v}^{(\mathcal{P}_{j_t})})$ to a common root for each $t = 1, 2, \dots, K$ and each $\vec{v}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}$, we obtain the following proposition.

Proposition 3.3.2. *Let $\bar{f}^{-1}(\{d\}) = \{d, j_1, j_2, \dots, j_K\}$. For $t = 1, 2, \dots, K$, let $\mathcal{P}_{j_t}$, m_{j_t} and $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{v}^{(\mathcal{P}_{j_t})})$ for $t = 1, 2, \dots, K$ be as in Proposition 3.3.1. Moreover, we denote by $\mathbf{0}$ the constant 0 function $\mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}$. Then*

$$\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q}) = \sum_{t=1}^{K} \sum_{\vec{v}^{(\mathcal{P}_{j_t})} \in \{\emptyset,\neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t},\mathbf{0}}(\vec{v}^{(\mathcal{P}_{j_t})}, (\emptyset, \dots, \emptyset)) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{v}^{(\mathcal{P}_{j_t})})^{+}).$$

In the third and final step, we consider vertices x from a *periodic* coset C_i (with $i < d$). Let us write $\bar{f}^{-1}(\{i\}) = \{i', j_1, j_2, \dots, j_K\}$, where i' is the unique $\bar{f}$-periodic pre-image of i under $\bar{f}$. Hence, C_{j_t} for $t = 1, 2, \dots, K$ is a transient coset; let $\mathcal{P}_{j_t}$ be the arithmetic partition of C_{j_t} defined in Proposition 3.3.1. Moreover, let $(i_0, i_1, \dots, i_{\ell-1})$ be the cycle of $i = i_0$ under $\bar{f}$, and let us define $i_k := i_{k \bmod \ell}$ for $k \in \mathbb{Z}$ (in particular, $i' = i_{-1}$). In addition, let $A_{i_k} : \mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}, x \mapsto \alpha_{i_k} x + \beta_{i_k}$, be the affine map that describes how f maps from C_{i_k} to $C_{i_{k+1}}$.

This case is more complicated, and we need to make a recursion by another parameter. As in Section 3.2, we denote by Γ_{per} the induced subgraph of Γ_f on the union of all periodic blocks C_i, i.e., the functional graph of f_{per}. We remind the reader that we explicitly understand the trees above periodic vertices in Γ_{per} thanks to Theorem 3.2.1. The idea is to proceed by recursion on a parameter called $\mathfrak{h}(x)$, which can range from 0 up to the maximum height H_i of the rooted trees in Γ_{per} above periodic vertices in one of the cosets $C_{i_t} = C_{\bar{f}^t(i)}$ for $t = 0, 1, \dots, \ell - 1$. This parameter is defined as follows:

$$\mathfrak{h}(x) = \begin{cases} \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(x)) \in \{0, 1, \dots, H_i - 1\}, & \text{if } x \text{ is } f\text{-transient}, \\ H_i, & \text{if } x \text{ is } f\text{-periodic}. \end{cases} \tag{3.1}$$

By Example 3.2.2, there is in general no relation between $\mathfrak{h}(x)$ and $\mathrm{ht}(\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(x))$ when $\mathfrak{h}(x) = H_i$ (i.e., when x is f-periodic). Also, $\mathfrak{h}$ need not assume all values in $\{0, 1, \dots, H_i\}$ on a given coset C_{i_t}; in fact, $\mathfrak{h}(C_{i_t})$ need not even be an integer interval. None of this will be an issue for our approach, though.

We observe that $H_i + 1$ is the smallest positive integer k such that for all $t \in \mathbb{Z}$, the common procreation number

$$\mathrm{proc}_k^{(\Gamma_{\mathrm{per}}^*)}(x) = \frac{\gcd(\prod_{j=0}^{k-1} \alpha_{i_{t-k+j}}, s)}{\gcd(\prod_{j=0}^{k-2} \alpha_{i_{t-k+j}}, s)}$$

of all f-periodic vertices $x \in C_{i_t}$ is equal to 1, which is (for each given t) equivalent to

$$\gcd\left(\prod_{j=0}^{k-1} \alpha_{i_{t-k+j}}, s\right) = \gcd\left(\prod_{j=0}^{k-2} \alpha_{i_{t-k+j}}, s\right),$$

and further to

$$\prod_{p \mid \gcd(\alpha_{i_{t-1}}, s)} p^{\nu_p(s)} \mid \prod_{j=0}^{k-2} \alpha_{i_{t-k+j}}.$$

Setting $\overline{\alpha}_i := \prod_{t=0}^{\ell-1} \alpha_{i_t}$ (the linear coefficient of $\mathcal{A}_i$ in the notation of Section 3.1), it is not difficult to see from this that

$$H_i \le \ell \cdot \max_{p \mid \gcd(\overline{\alpha}_i, s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i)} \right\rceil \le \ell \operatorname{mpe}(s) \le d \operatorname{mpe}(q-1). \tag{3.2}$$

Let us set $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$ for $t = 1, 2, \ldots, K$, and $\mathcal{R}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$. We denote by n_i the length of the spanning congruence sequence for $\mathcal{R}_i$ which we use (in general, $n_i = \sum_{t=1}^{K} (m_{j_t} + 1)$, but in a concrete example there may be repetitions among those congruences, allowing us to delete some of them). A simple observation is that as far as the transient coset contribution $\operatorname{Tree}_{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$ to $\operatorname{Tree}_{\Gamma_f}(x)$ is concerned, everything is as in step 1.

Proposition 3.3.3. *Let $i \in \{0, 1, \ldots, d-1\}$ be $\bar{f}$-periodic, and let $j_1, j_2, \ldots, j_K$ be the $\bar{f}$-transient pre-images of i under $\bar{f}$. Moreover, let $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$ for $t = 1, 2, \ldots, K$ and $\mathcal{R}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$. Then the following hold.*

(1) *For $x \in C_i$ and $t \in \{1, 2, \ldots, K\}$, the isomorphism type $\operatorname{Tree}_{\Gamma_f}(x, C_{j_t})$ only depends on the $\mathcal{P}'_{j_t}$-block $\mathcal{B}(\mathcal{P}'_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ (for some $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$) in which x lies. That isomorphism type is denoted by $\operatorname{Tree}_i(\mathcal{P}'_{j_t}, C_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ and can be computed according to the formula*

$$\begin{aligned}&\operatorname{Tree}_i\left(\mathcal{P}'_{j_t}, C_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right)\\ &\quad= \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right) \operatorname{Tree}_{j_t}\left(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})}\right)^{+}.\end{aligned}$$

(2) *For $x \in C_i$, the rooted tree isomorphism type of $\operatorname{Tree}_{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$ only depends on the $\mathcal{R}_i$-block $\mathcal{B}(\mathcal{R}_i, \vec{\nu}^{(\mathcal{R}_i)})$ (for some $\vec{\nu}^{(\mathcal{R}_i)} \in \{\emptyset, \neg\}^{n_i}$) in which x lies. That isomorphism type is denoted by $\operatorname{Tree}_i(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{R}_i)})$ and can be computed according to the formula*

$$\operatorname{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{R}_i)}\right) = \sum_{t=1}^{K} \operatorname{Tree}_i\left(\mathcal{P}'_{j_t}, C_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right),$$

where $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$*, for* $t = 1, 2, \ldots, K$*, is the unique logical sign tuple such that* $\mathcal{B}(\mathcal{R}_i, \vec{\nu}^{(\mathcal{R}_i)}) \subseteq \mathcal{B}(\mathcal{P}'_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$*; in the standard situation, where* $n_i = \sum_{t=1}^{K} (m_{j_t} + 1)$*, the tuple* $\vec{\nu}^{(\mathcal{R}_i)}$ *is simply the concatenation of the* $\vec{\nu}^{(\mathcal{P}'_{j_t})}$ *for* $t = 1, 2, \ldots, K$*.*

We remind the reader that we wish to proceed by recursion on the parameter $\mathfrak{h}(x)$ defined in (3.1). This is motivated by Proposition 3.3.3, because if $\mathfrak{h}(x) = 0$, then x has no transient children in $C_{i'}$, whence

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}\left(x, \bigcup_{t=1}^{K} C_{j_t}\right).$$

In general, we wish to construct an arithmetic partition $\mathcal{S}_{i,h}$ of C_i such that for vertices $x \in C_i$ with $\mathfrak{h}(x) = h$, the isomorphism type of $\mathrm{Tree}_{\Gamma_f}(x, C_{i'})$ only depends on the $\mathcal{S}_{i,h}$-block containing x and is explicitly understood. Then we are basically done, because

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}\left(x, \bigcup_{t=1}^{K} C_{j_t}\right) + \mathrm{Tree}_{\Gamma_f}(x, C_{i'}).$$

In order to construct $\mathcal{S}_{i,h}$ and prove that it has the desired property, we need to introduce quite a few notations.

- For $h \in \{0, 1, \ldots, H_i\}$, a vertex $x \in C_i = C_{i_0}$ has at least h successor generations in Γ^*_{per} if and only if x lies in the image of $A_{i_{-h}} A_{i_{-h+1}} \cdots A_{i_{-1}}$, which is the affine map $\mathcal{A}_{i,h} : z \mapsto \overline{\alpha}_{i,h} z + \overline{\beta}_{i,h}$ of $\mathbb{Z}/s\mathbb{Z}$, where

$$\overline{\alpha}_{i,h} = \prod_{t=1}^{h} \alpha_{i_{-t}} \quad \text{and} \quad \overline{\beta}_{i,h} = \sum_{t=1}^{h} \beta_{i_{-t}} \prod_{k=1}^{t-1} \alpha_{i_{-k}}.$$

  That is, x has at least h successor generations in Γ^*_{per} if and only if it satisfies the following s-congruence, which we denote by $\theta_{i,h}(x)$:

$$x \equiv \overline{\beta}_{i,h} \pmod{\gcd(\overline{\alpha}_{i,h}, s)}.$$

  We observe that the modulus in $\theta_{i,0}(x)$ is 1, so that congruence is trivial.

- Next, we describe, for each $h \in \{0, 1, \ldots, H_i\}$, a simple system $\Theta_{i,h}(x)$ of at most two s-CCs such that for all $x \in C_i$, the equality $\mathfrak{h}(x) = h$ holds if and only if $\Theta_{i,h}(x)$ holds. If $h < H_i$, then the vertices $x \in C_i$ with $\mathfrak{h}(x) = h$ are just those f-transient $x \in C_i$ that have exactly h successor generations in Γ^*_{per}. It follows that for such h, one has $\mathfrak{h}(x) = h$ if and only if $\theta_{i,h}(x)$ and $\neg\theta_{i,h+1}(x)$ both hold. Now we assume that $h = H_i$. By definition of $\mathfrak{h}$, this happens if and only if x is f-periodic, which is (by definition of H_i) equivalent to x having at least H_i

successor generations in Γ^*_{per}. Therefore, the condition $\theta_{i,H_i}(x)$ alone provides the desired characterization in this case.
Now, noting once more that $\theta_{i,0}(x)$ is trivial and may thus be omitted from any system of conditions containing it, we may define $\Theta_{i,h}(x)$ as follows:

$$\Theta_{i,h}(x) := \begin{cases} \emptyset, & \text{if } H_i = (h =)0, \\ \neg\theta_{i,1}(x), & \text{if } h = 0 < H_i, \\ \theta_{i,h}(x) \wedge (\neg\theta_{i,h+1}(x)), & \text{if } 0 < h < H_i, \\ \theta_{i,H_i}(x), & \text{if } h = H_i > 0. \end{cases}$$

- If $\Theta(x)$ is a system of m-CCs, then $\mathfrak{P}(\Theta(x))$ denotes the arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ spanned by the non-negated versions of the conditions in $\Theta(x)$. For example,

$$\mathfrak{P}\begin{pmatrix} x \equiv 4 \pmod 6 \\ x \not\equiv 3 \pmod 9 \\ x \equiv 0 \pmod 2 \end{pmatrix} = \mathfrak{P}\begin{pmatrix} x \equiv 4 \pmod 6 \\ x \equiv 3 \pmod 9 \\ x \equiv 0 \pmod 2 \end{pmatrix}.$$

We note that if $x \in \mathbb{Z}/m\mathbb{Z}$ is chosen from a fixed block of $\mathfrak{P}(\Theta(x))$, then the truth value of each condition in $\Theta(x)$ is independent of x, and so is the truth value of $\Theta(x)$ itself. We also observe that for each $h \in \{0, 1, \ldots, H_i\}$, the following equality holds for the systems $\Theta_{i,k}(x)$ of s-CCs defined in the previous bullet point:

$$\bigwedge_{k=0}^{h} \mathfrak{P}(\Theta_{i,k}(x)) = \mathfrak{P}(\theta_{i,j}(x) : j = 1, 2, \ldots, \min\{h+1, H_i\}). \tag{3.3}$$

We set $\mathcal{U}_i := \mathfrak{P}(\theta_{i,k}(x) : k = 1, 2, \ldots, H_i)$.

- For $k \in \{0, 1, \ldots, H_i\}$, we denote by $\vec{\xi}_{i,k}$ the logical sign tuple $(\nu_1, \nu_2, \ldots, \nu_{H_i})$ with $\nu_t = \emptyset$ if and only if $t \leq k$. With this definition, if we view $\mathcal{U}_i$ as an arithmetic partition of C_i, then the block $\mathcal{B}(\mathcal{U}_i, \vec{\xi}_{i,k})$ consists precisely of those $x \in C_i$ such that $\mathfrak{h}(x) = k$ (and every block of $\mathcal{U}_i$ is of this form for some k).
- Let $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$ be an arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$, and let $A : z \mapsto az + b$, be an affine map of $\mathbb{Z}/m\mathbb{Z}$. In dependency of $\mathcal{P}$ (actually, of the fixed sequence of spanning congruences for $\mathcal{P}$, rather than $\mathcal{P}$ itself) and A, we define another arithmetic partition $\lambda(\mathcal{P}, A)$ of $\mathbb{Z}/m\mathbb{Z}$ as follows: $\lambda(\mathcal{P}, A) := \mathfrak{P}(x \equiv a\mathfrak{b}_j + b \pmod{\gcd(a\mathfrak{a}_j, m)} : j = 1, 2, \ldots, K)$. Here is a list of important facts concerning this notation (for which we have $m = s$ throughout):
  - If $\mathcal{P}$ is an arithmetic partition of $C_{i'}$, then

$$\mathfrak{P}'(\mathcal{P}, A_{i'}) = \lambda(\mathcal{P}, A_{i'}) \wedge \mathfrak{P}(\theta_{i,1}(x)).$$

– For $h = 0, 1, \ldots, H_{i'} = H_i$, we have

$$\lambda(\mathfrak{P}(\theta_{i',h}(x)), A_{i'}) = \begin{cases} \mathfrak{P}(\theta_{i,h+1}(x)), & \text{if } h < H_i, \\ \mathfrak{P}(\theta_{i,H_i}(x)), & \text{if } h = H_i. \end{cases}$$

Indeed, for $h < H_i$, this is immediate by the definition of $\theta_{i,h}(x)$ and the facts that $A_{i'}(\overline{\beta}_{i',h}) = \overline{\beta}_{i,h+1}$ and $\gcd(a\gcd(a', s), s) = \gcd(aa', s)$ for all $a, a' \in \mathbb{Z}$. Moreover, noting that our definition of $\theta_{i,h}(x)$ also makes sense if $h > H_i$, we have $\lambda(\mathfrak{P}(\theta_{i',H_{i'}}(x))) = \mathfrak{P}(\theta_{i,H_i+1}(x))$. However, for $x \in C_i$, the condition $\theta_{i,H_i+1}(x)$ holds if and only if x has at least $H_i + 1$ successor generations in Γ^*_{per}, which is the case if and only if x is periodic, i.e., if and only if $\theta_{i,H_i}(x)$ holds. Hence, the congruences $\theta_{i,H_i+1}(x)$ and $\theta_{i,H_i}(x)$ have the same solution set in $\mathbb{Z}/s\mathbb{Z}$, whence $\mathfrak{P}(\theta_{i,H_i+1}(x)) = \mathfrak{P}(\theta_{i,H_i}(x))$. This concludes the proof of the above formulas for $\lambda(\mathfrak{P}(\theta_{i',h}(x)), A_{i'})$.

– By the previous two bullet points and equality (3.3), applied with i' in place of i, we have that

$$\mathfrak{P}'\Big(\bigwedge_{k=0}^{h} \mathfrak{P}(\Theta_{i',k}(x)), A_{i'}\Big) = \bigwedge_{k=0}^{\min\{h+1,H_i\}} \mathfrak{P}(\Theta_{i,k}(x)),$$

and, in particular, $\mathfrak{P}'(\mathcal{U}_{i'}, A_{i'}) = \mathcal{U}_i$.

- Let $\mathcal{P}$ be an arithmetic partition of C_i. We define the notation $\lambda_i^t(\mathcal{P})$, where t is a non-negative integer, as follows recursively: $\lambda_i^0(\mathcal{P}) := \mathcal{P}$, and for $t \geq 1$, we set $\lambda_i^t(\mathcal{P}) := \lambda(\lambda_i^{t-1}(\mathcal{P}), A_{i_{t-1}})$. In other words, $\lambda_i^t(\mathcal{P})$ is the arithmetic partition of C_{i_t} obtained by pushing $\mathcal{P}$ forward t times along the $\bar{f}$-cycle of i via the operation λ, using the appropriate affine function A_{i_k} in each step.
- For $h \in \{0, 1, \ldots, H_i\}$, we introduce the following arithmetic partitions of C_i:
  - $\mathcal{S}_{i,h} := \bigwedge_{t=1}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}})$;
  - $\mathcal{P}_{i,h} := \mathcal{R}_i \wedge \mathcal{S}_{i,h} = \bigwedge_{t=0}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}})$;
  - $\mathcal{T}_{i,h} := \mathcal{S}_{i,h} \wedge \mathcal{U}_i = \bigwedge_{t=1}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}}) \wedge \mathcal{U}_i$;
  - $\mathcal{Q}_{i,h} := \mathcal{R}_i \wedge \mathcal{T}_{i,h} = \bigwedge_{t=0}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}}) \wedge \mathcal{U}_i$.

  The motivation for considering $\mathcal{S}_{i,h}$ and $\mathcal{P}_{i,h}$ is that their blocks control the rooted tree isomorphism type of $\mathrm{Tree}_{\Gamma_f}(x, C_{i'})$ and $\mathrm{Tree}_{\Gamma_f}(x)$, respectively, for vertices $x \in C_i$ *of* $\mathfrak{h}$*-value* h contained in them – see Proposition 3.3.4 below. An explicit formula for $\mathrm{Tree}_{\Gamma_f}(x, C_{i'})$ in terms of the $\mathcal{S}_{i,h}$-block containing x is also given in Proposition 3.3.4, and that formula involves distribution numbers (as in Lemma 2.2.2 – see the sentence after that lemma) of the partitions $\mathcal{Q}_{i',k}$ for $0 \leq k < h$. The partitions of the form $\mathcal{T}_{i,h}$ are not mentioned in the statement of

Proposition 3.3.4, but they play an important role in its proof due to the fact that for $h \geq 1$, one has $\mathcal{T}_{i,h} = \mathfrak{P}'(\mathcal{Q}_{i',h-1}, A_{i'})$.

- We denote the concatenation of logical sign tuples $\vec{\nu}$ and $\vec{\nu}'$ by $\vec{\nu} \diamond \vec{\nu}'$.

We are now in a position to formulate in detail how the blocks B of $\mathcal{S}_{i,h}$ affect the rooted trees above vertices $x \in B$ with $\mathfrak{h}(x) = h$.

Proposition 3.3.4. *Let $i \in \{0, 1, \ldots, d-1\}$ be $\bar{f}$-periodic, with $\bar{f}$-pre-images $i', j_1, j_2, \ldots, j_K$, where i' is $\bar{f}$-periodic. Moreover, let $h \in \{0, 1, \ldots, H_i\}$, and let $x \in C_i$ with $\mathfrak{h}(x) = h$. Then the following hold.*

(1) *The isomorphism type $\mathrm{Tree}_{\Gamma_f}(x, C_{i'})$ only depends on the $\mathcal{S}_{i,h}$-block $\mathcal{B}(\mathcal{S}_{i,h}, \vec{\nu}^{(\mathcal{S}_{i,h})})$ in which x lies and is denoted by $\mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$.*

(2) *The isomorphism type $\mathrm{Tree}_{\Gamma_f}(x)$ depends on $\mathcal{B}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$, the $\mathcal{P}_{i,h}$-block in which x lies and is denoted by $\mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$.*

More specifically, for $h = 0$, where $\mathcal{S}_{i,h} = \mathcal{P}(\emptyset)$ and $\mathcal{P}_{i,h} = \mathcal{R}_i$, the rooted tree $\mathrm{Tree}_i^{(0)}(\mathcal{S}_{i,0}, C_{i'}, \emptyset)$ is trivial, and

$$\mathrm{Tree}_i^{(0)}(\mathcal{P}_{i,0}, \vec{\nu}^{(\mathcal{P}_{i,0})}) = \mathrm{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{P}_{i,0})}\right).$$

For $h \geq 1$, writing $\vec{\nu}^{(\mathcal{S}_{i,h})} = \diamond_{j=1}^{h} \vec{o}'_t$ with $\vec{o}'_t \in \{\emptyset, \neg\}^{n_{i-t}}$, we have the following, where $\vec{o}_t$ for $t = 0, 1, \ldots, h-1$ is a variable ranging over $\{\emptyset, \neg\}^{n_{i-t-1}}$:

$$\begin{aligned}&\mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})\\ &= \sum_{k=0}^{h-1} \sum_{\vec{o}_0, \ldots, \vec{o}_k} \sigma_{\mathcal{Q}_{i',k}, A_{i'}}(\diamond_{t=0}^{k} \vec{o}_t \diamond \vec{\xi}_{i',k}, \diamond_{t=1}^{k+1} \vec{o}'_t \diamond \vec{\xi}_{i,h})\, \mathrm{Tree}_{i'}^{(k)}(\mathcal{P}_{i',k}, \diamond_{t=0}^{k} \vec{o}_t)^{+},\end{aligned}$$

and, viewing $\vec{\nu}^{(\mathcal{P}_{i,h})}$ as the concatenation of $\vec{o}'_0 \in \{\emptyset, \neg\}^{n_i}$ and $\vec{\nu}^{(\mathcal{S}_{i,h})}$, we have

$$\mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})}) = \mathrm{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{o}'_0\right) + \mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})}).$$

Proof. The formulas for $h = 0$ are clear because vertices $x \in C_i$ with $\mathfrak{h}(x) = 0$ have no f-transient children in $C_{i'}$. We may thus assume that $h \geq 1$. With regard to $\mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$, we note that if $x \in C_i$ lies in the block $\mathcal{B}(\mathcal{S}_{i,h}, \vec{\nu}^{(\mathcal{S}_{i,h})})$ of $\mathcal{S}_{i,h}$ and satisfies $\mathfrak{h}(x) = h$, then for each $k \in \{0, 1, \ldots, h-1\}$, we have that x lies in the block $\mathcal{B}(\mathcal{T}_{i,k+1}, \diamond_{j=1}^{k+1} \vec{o}'_j \diamond \vec{\xi}_{i,h})$ of $\mathcal{T}_{i,k+1}$. By definition, $\mathcal{T}_{i,k+1} = \mathcal{P}'(\mathcal{Q}_{i',k}, A_{i'})$. Each ($f$-transient) pre-image y of x in $C_{i'}$ with $\mathfrak{h}(y) = k$ lies in some block of $\mathcal{Q}_{i',k}$ of the form $\mathcal{B}(\mathcal{Q}_{i',k}, \diamond_{j=0}^{k} \vec{o}_j \diamond \vec{\xi}_{i',k})$, where $\vec{o}_j \in \{\emptyset, \neg\}^{n_{i-j-1}}$ for $j = 0, 1, \ldots, k$.

Moreover, by Lemma 2.2.2, the number of such pre-images in that block is

$$\sigma_{\mathcal{Q}_{i',k},A_{i'}}(\diamond_{j=0}^{k}\vec{o}_j \diamond \vec{\xi}_{i',k}, \diamond_{j=1}^{k+1}\vec{o}'_j \diamond \vec{\xi}_{i,h}).$$

Each pre-image y of x in $C_{i'}$ which is contained in $\mathcal{B}(\mathcal{Q}_{i',k}, \diamond_{j=0}^{k}\vec{o}_j \diamond \vec{\xi}_{i',k})$ is also contained in $\mathcal{B}(\mathcal{P}_{i',k}, \diamond_{j=0}^{k}\vec{o}_j)$, whence

$$\mathrm{Tree}_{\Gamma_f}(y) \cong \mathrm{Tree}_{i'}^{(k)}(\mathcal{P}_{i',k}, \diamond_{j=0}^{k}\vec{o}_j).$$

This concludes the proof of the formula for $\mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{v}^{(\mathcal{S}_{i,h})})$.

The formula for $\mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{v}^{(\mathcal{P}_{i,h})})$ is clear because

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}\left(x, \bigcup_{t=1}^{K} C_{j_t}\right) + \mathrm{Tree}_{\Gamma_f}(x, C_{i'})$$

and, by Proposition 3.3.3 (2),

$$\mathrm{Tree}_{\Gamma_f}\left(x, \bigcup_{t=1}^{K} C_{j_t}\right) = \mathrm{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{o}'_0\right)$$

for all $x \in C_i$. ∎

Now, let us set

$$\mathcal{P}_i := \mathcal{Q}_{i,H_i} = \mathcal{P}_{i,H_i} \wedge \mathcal{U}_i.$$

Putting everything together, we obtain the following concluding result for this section.

Proposition 3.3.5. *Let $i \in \{0, 1, \ldots, d-1\}$ be $\bar{f}$-periodic, and let $\mathcal{B}(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ be a block of $\mathcal{P}_i$. We can view $\vec{v}^{(\mathcal{P}_i)}$ as the concatenation $\diamond_{t=0}^{H_i}\vec{o}'_t \diamond \vec{\xi}$, where $\vec{o}'_t \in \{\emptyset, \neg\}^{n_{i-t}}$ for $t = 0, 1, \ldots, H_i$, and $\vec{\xi} = \vec{\xi}_{i,h}$ for a unique $h \in \{0, 1, \ldots, H_i\}$. Then for $x \in \mathcal{B}(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$, the isomorphism type of $\mathrm{Tree}_{\Gamma_f}(x)$ does not depend on x, is denoted by $\mathrm{Tree}_i(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ and given by the formula*

$$\mathrm{Tree}_i(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)}) = \mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \diamond_{t=0}^{h}\vec{o}'_t).$$

Moreover, we have

$$\mathrm{AC}(\mathcal{P}_i) \leq d^2\,\mathrm{mpe}(q-1) + d - 1 \in O(d^2\,\mathrm{mpe}(q-1)).$$

Proof. The block $\mathcal{B}(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ is contained in $\mathcal{B}(\mathcal{U}_i, \xi_{i,h})$, whence $\mathfrak{h}(v) = h$. Additionally, $\mathcal{B}(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ is contained in $\mathcal{B}(\mathcal{P}_{i,h}, \diamond_{t=0}^{h}\vec{o}'_t)$, so the asserted formula for $\mathrm{Tree}_i(\mathcal{P}_i, \vec{v}^{(\mathcal{P}_i)})$ is clear by Proposition 3.3.4 (2). For the complexity bound, we note

that by definition of $\mathcal{P}_i = \mathcal{P}_{i,H_i} \wedge \mathcal{U}_i$ and the bound (3.2), we have

$$\begin{aligned}\mathrm{AC}(\mathcal{P}_i) &\le \mathrm{AC}(\mathcal{P}_{i,H_i}) + \mathrm{AC}(\mathcal{U}_i)\\ &\le \sum_{t=0}^{H_i} \mathrm{AC}(\lambda_{i-t}^{t}(\mathcal{R}_{i-t})) + H_i \le (H_i+1)(d-1) + H_i\\ &= H_i d + d - 1 \le d^2 \operatorname{mpe}(q-1) + d - 1,\end{aligned}$$

as required. ∎

Remark 3.3.6. Omitting all explicit details which we worked out in this section, we basically proved that on each coset C_i, there is an arithmetic partition $\mathcal{P}_i$ with $\mathrm{AC}(\mathcal{P}_i) \in O(d^2 \operatorname{mpe}(q-1)) \subseteq O(d^2 \log q)$ which "controls" the rooted trees above vertices in C_i. Now, the trivial partition $\mathcal{T}_s$ of $C_i \cong \mathbb{Z}/s\mathbb{Z}$ which consists entirely of singleton blocks also "controls" the trees above its blocks, for trivial reasons. While it is preferable for our effective purposes to subsume as many isomorphic rooted trees under a common block as possible (and thus $\mathcal{P}_i$ is in general preferable over $\mathcal{T}_s$), it is an interesting question whether $\mathcal{T}_s$ could "beat" $\mathcal{P}_i$ at least as far as arithmetic complexity is concerned.

Let us discuss this problem for a general modulus $m \in \mathbb{N}^+$ (not just $s=(q-1)/d$). We consider the factorization $m = p_1^{v_1} p_2^{v_2} \cdots p_K^{v_K}$ of m into pairwise coprime prime powers. By adding logical signs to the m-CCs in the system consisting of

$$x \equiv b \pmod{p_j^{v_j}}$$

for $j = 1, 2, \ldots, K$ and $b \in \{0, 1, \ldots, p_j^{v_j} - 2\}$, one can obtain each singleton subset of $\mathbb{Z}/m\mathbb{Z}$ as a block of the corresponding arithmetic partition. This shows that $\mathrm{AC}(\mathcal{T}_m) \le p_1^{v_1} + \cdots + p_K^{v_K} - K$. Of course, if m is a prime power, then this bound is just $m-1$. On the other hand, if $m = p_1 p_2 \cdots p_K = p_K\#$ is a primorial, then $p_K\# = \exp((1+o(1))K \log K)$, and thus $\log m \sim K \log K$, i.e., recalling from Remark 1.6 that W denotes the Lambert W function, we have

$$K \sim \frac{\log m}{W(\log m)} \sim \frac{\log m}{\log\log m}.$$

Our bound implies that $\mathrm{AC}(\mathcal{T}_m)$ is at most

$$\begin{aligned}p_1 + \cdots + p_K - K &\sim p_1 + \cdots + p_K \sim \frac{1}{2} K^2 \log K\\ &\sim \frac{1}{2} \frac{\log^2 m}{\log\log^2 m} (\log\log m - \log\log\log m) \sim \frac{1}{2} \frac{\log^2 m}{\log\log m},\end{aligned}$$

which does not beat $O(d^2 \log m)$ for fixed d, and even less so $O(d^2 \operatorname{mpe}(m)) = O(d^2)$, noting that $\operatorname{mpe}(m) = \operatorname{mpe}(p_K\#) = 1$. It is an interesting open question

whether

$$\liminf_{m\to\infty} \frac{\mathrm{AC}(\mathcal{T}_m)}{\log^2(m)/\log\log m} > 0,$$

see also Question 6.3.1.

3.4 Understanding the connected components

Now we want to combine the theory developed thus far to understand the connected components of Γ_f in their entirety. From the introduction, we recall our approach of associating a necklace of rooted tree isomorphism types with each connected component of Γ_f, which characterizes the digraph isomorphism type of that component.

Let $\mathcal{L}$ be a CRL-list of f (see Section 3.1 on how to construct $\mathcal{L}$). We remind the reader that the first entries of the pairs in $\mathcal{L}$ are representatives not only for the cycles of f, but also for the connected components of Γ_f. Let us fix $(r, l) \in \mathcal{L}$. We aim to give a neat description of the cyclic sequence of rooted tree isomorphism types associated with the connected component of Γ_f containing r.

We note that by our construction of $\mathcal{L}$, if $r = 0_{\mathbb{F}_q}$, then $l = 1$, and the connected component consists of a single rooted tree attached to the looped vertex $0_{\mathbb{F}_q}$. We can determine that tree, $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, as described in Section 3.3. The length 1 cyclic sequence $[\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})]$ determines the connected component of $0_{\mathbb{F}_q}$ as a whole, and so we may henceforth assume that $r \neq 0_{\mathbb{F}_q}$, contained in a unique coset C_i of C in $\mathbb{F}_q^*$.

We recall that i is necessarily a periodic point of $\bar{f}$, and let $(i_0, i_1, \ldots, i_{\ell-1})$ with $i_0 = i$ be its cycle. For general $t \in \mathbb{Z}$, we set $i_t := i_{t \bmod \ell}$. By Section 3.3, on each coset C_j of C in $\mathbb{F}_q^*$, we have an arithmetic partition $\mathcal{P}_j$ such that the isomorphism type of $\mathrm{Tree}_{\Gamma_f}(x)$ is constant and explicitly understood for vertices $x \in C_j$ chosen from a common block $\mathcal{B}(\mathcal{P}_j, \vec{v})$ of $\mathcal{P}_j$, and we denote the said isomorphism type by $\mathrm{Tree}_j(\mathcal{P}_j, \vec{v})$.

Let us fix $t \in \{0, 1, \ldots, \ell - 1\}$ and recall (from the previous section) the notation $\mathcal{A}_{i_t} = A_{i_t} A_{i_{t+1}} \cdots A_{i_{t+\ell-1}}$ for the product of all affine maps along the cycle of i_t. Also, we recall that f^ℓ stabilizes C_{i_t}, and that the restriction $(f^\ell)_{|C_{i_t}}$ corresponds to the affine map $\mathcal{A}_{i_t}$ of $\mathbb{Z}/s\mathbb{Z}$. Let us set $r'_{i_t} := f^t(r)$. We have $\ell \mid l$, and the cycle of r'_{i_t} under $\mathcal{A}_{i_t}$ is

$$\begin{aligned}&(r'_{i_t}, \mathcal{A}_{i_t}(r'_{i_t}), \mathcal{A}_{i_t}^2(r'_{i_t}), \ldots, \mathcal{A}_{i_t}^{l/\ell-1}(r'_{i_t}))\\ &\quad = (f^t(r), f^{t+\ell}(r), f^{t+2\ell}(r), \ldots, f^{t+((l/\ell)-1)\ell}(r)).\end{aligned}$$

We denote by $\mathcal{B}_{i_t}$ the function defined on C_{i_t} that maps $x \in C_{i_t}$ to the unique block of $\mathcal{P}_{i_t}$ containing x. Our next goal is to understand the block sequence

$$(\mathcal{B}_{i_t}(r'_{i_t}), \mathcal{B}_{i_t}(\mathcal{A}_{i_t}(r'_{i_t})), \ldots, \mathcal{B}_{i_t}(\mathcal{A}_{i_t}^{l/\ell-1}(r'_{i_t})))$$

along the $\mathcal{A}_{i_t}$-cycle of r'_{i_t} for each $t = 0, 1, \ldots, \ell - 1$, because those sequences combine to the block sequence $(\mathcal{B}_{i_n}(f^n(r)))_{n=0,1,\ldots,l-1}$, from which one can read off the cyclic sequence of rooted tree isomorphism types for the connected component of Γ_f containing r.

Now, let us assume that $\mathcal{P}_{i_t} = \mathfrak{P}(x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}} : j = 1, 2, \ldots, m_{i_t})$. Understanding the block sequence $(\mathcal{B}_{i_t}(\mathcal{A}_{i_t}^m(r'_{i_t})))_{m=0,1,\ldots,l/\ell-1}$ means understanding the truth values of the congruences $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$ as x ranges over the cycle of r'_{i_t} under $\mathcal{A}_{i_t}$. We recall from the previous section that $\overline{\alpha}_{i_t} := \prod_{t=0}^{\ell-1} \alpha_{i_t}$ is the linear coefficient of $\mathcal{A}_{i_t}$. Lemma 2.1.14 implies that all periodic points of $\mathcal{A}_{i_t}$ (in particular all points on the cycle of r'_{i_t} under $\mathcal{A}_{i_t}$) have one particular, explicitly known value modulo $\prod_{p|\gcd(\overline{\alpha}_{i_t},s)} p^{v_p(s)}$. This may cause some of the congruences $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$ for $j \in \{1, 2, \ldots, m_{i_t}\}$ to have constant truth value on all periodic points of $\mathcal{A}_{i_t}$. The remaining congruences can be dealt with as follows.

We compute $\log_{\mathcal{A}_{i_t}}^{(\mathfrak{a}_{i_t,j})}(r'_{i_t}, \mathfrak{b}_{i_t,j}) =: \mathfrak{l}_{i_t,j}$ (see Section 2.4 for this discrete log notation) and the cycle length $l_{i_t,j}$ of r'_{i_t} under $\mathcal{A}_{i_t}$ modulo $\mathfrak{a}_{i_t,j}$. If $\mathfrak{l}_{i_t,j} = \infty$ (i.e., $\mathfrak{b}_{i_t,j}$ does not lie on the cycle of r'_{i_t} under $\mathcal{A}_{i_t}$ modulo $\mathfrak{a}_{i_t,j}$), then the congruence $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$ is false for all x on the $\mathcal{A}_{i_t}$-cycle of r'_{i_t} modulo s. Otherwise, that congruence is true precisely for those $x = \mathcal{A}_{i_t}^y(r'_{i_t})$ for which $y \equiv \mathfrak{l}_{i_t,j} \pmod{l_{i_t,j}}$. Let us denote by I_{i_t} the set of those $j \in \{1, 2, \ldots, m_{i_t}\}$ for which the truth value of $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$ is constant along the $\mathcal{A}_{i_t}$-cycle of r'_{i_t} modulo s. The block sequence $(\mathcal{B}_{i_t}(\mathcal{A}_{i_t}^m(r'_{i_t})))_{m=0,1,\ldots,l/\ell-1}$ is determined by

- the information about the constant truth values along the $\mathcal{A}_{i_t}$-cycle of r'_{i_t} modulo s of the congruences $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$ for $j \in I_{i_t}$, and
- the arithmetic partition $\mathcal{P}^{(i_t)} := \mathfrak{P}(y \equiv \mathfrak{l}_{i_t,j} \pmod{l_{i_t,j}} : j \notin I_{i_t})$ of $\mathbb{Z}/(l/\ell)\mathbb{Z}$, which encodes the behavior of the truth values of the remaining congruences $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$, for $j \notin I_{i_t}$, along the cycle.

Once the block sequence $(\mathcal{B}_{i_t}(\mathcal{A}_{i_t}^m(r'_{i_t})))_{m=0,1,\ldots,l/\ell-1}$ has been understood that way for each t, the actual necklace that encodes the isomorphism type of the connected component of Γ_f containing r is given by the cyclic sequence

$$[\mathrm{Tree}_{i_n}(\mathcal{P}_{i_n}, \mathcal{B}_{i_n}(\mathcal{A}_{i_n}^{(n-(n \bmod \ell))/\ell}(r'_{i_n})))]_{n=0,1,\ldots,l-1}, \tag{3.4}$$

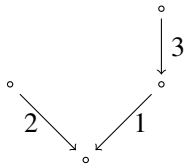
where, by abuse of notation, $\mathrm{Tree}_i(\mathcal{P}_i, B)$ is to be understood as $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu})$ for $B = \mathcal{B}(\mathcal{P}_i, \vec{\nu})$.

The connected components of Γ_f associated with two different choices for r are isomorphic if and only if the corresponding cyclic sequences (3.4) are equal. We do note, however, that it does not appear obvious how to check this efficiently, just as it does not seem clear how to check efficiently whether two given arithmetic partitions are equal – see Problems 6.2.3 and 6.2.4.

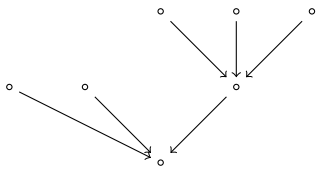
Chapter 4

Computations and examples

The goal of this chapter is to illustrate the theory developed thus far through some concrete computations and examples/special cases. We start by introducing a useful notation that is used in Sections 4.1 and 4.2. We consider finite directed rooted trees (with all arcs oriented toward the root) that have non-negative integers as edge weights. An *isomorphism* of such graphs is one of the underlying non-edge-weighted directed graphs that preserves the weights of arcs. With each isomorphism type $\mathfrak{T}$ of such trees, we associate an isomorphism type Expand($\mathfrak{T}$) of non-edge-weighted, finite directed rooted trees as follows. If $y_1, y_2, \ldots, y_n$ are the neighbors of the root x of $\mathfrak{T}$, and they have the (isomorphism types of) edge-weighted rooted trees $\mathfrak{T}_1, \mathfrak{T}_2, \ldots, \mathfrak{T}_n$ attached to them and the edge joining x and y_j has weight $w_j \in \mathbb{N}_0$, then Expand($\mathfrak{T}$) is defined recursively by taking a new root and attaching w_j copies of Expand($\mathfrak{T}_j$) to it for each $j = 1, 2, \ldots, n$. For example, if $\mathfrak{T}$ is



then Expand($\mathfrak{T}$) is



By recursion on the height of $\mathfrak{T}$, we define $\mathfrak{T}$ to be *simplified* as follows. The trivial isomorphism type $\mathfrak{T}$ is simplified, and if $\mathfrak{T}$ is of positive height, then $\mathfrak{T}$ is simplified if all isomorphism types $\mathfrak{T}'$ attached to the root of $\mathfrak{T}$ (which are all of smaller height than $\mathfrak{T}$) are simplified, pairwise distinct, and none of them is attached to the root with edge weight 0. In fact, we could have omitted 0 as an edge weight from the start, but it is more convenient to include it because in the subsequent discussion, the weights sometimes are formulas involving greatest common divisors that may simplify to 0 under certain assumptions.

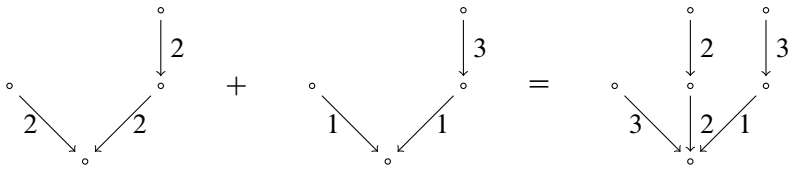
The simplified isomorphism types of finite edge-weighted directed rooted trees are in bijection with the isomorphism types of finite directed rooted trees via Expand (as can be easily proved by induction on the height). This allows us to define the

simplified form $\mathrm{SF}(\mathfrak{T})$ of an arbitrary isomorphism type $\mathfrak{T}$ of finite edge-weighted directed rooted trees as the unique simplified isomorphism type such that

$$\mathrm{Expand}(\mathrm{SF}(\mathfrak{T})) = \mathrm{Expand}(\mathfrak{T}).$$

We write $\mathfrak{T} \sim \mathfrak{T}'$ for $\mathrm{SF}(\mathfrak{T}) = \mathrm{SF}(\mathfrak{T}')$ (equivalently, $\mathrm{Expand}(\mathfrak{T}) = \mathrm{Expand}(\mathfrak{T}')$). The simplified form of $\mathfrak{T}$ can be constructed explicitly from $\mathfrak{T}$ in a simple recursion on the tree height (going through the $\mathfrak{T}'$ attached to the root of $\mathfrak{T}$, computing their simplified forms, and adding up edge weights that belong to the same $\mathrm{SF}(\mathfrak{T}')$).

In Section 3.3, we introduced a sum of isomorphism types of non-edge-weighted finite directed rooted trees (turning their class into a class-sized monoid), and there is a unique way to define a *sum of simplified edge-weighted rooted tree isomorphism types* such that Expand becomes a monoid isomorphism (i.e., $\mathrm{Expand}(\mathfrak{T}_1 + \mathfrak{T}_2) = \mathrm{Expand}(\mathfrak{T}_1) + \mathrm{Expand}(\mathfrak{T}_2)$). Explicitly, $\mathfrak{T}_1 + \mathfrak{T}_2$ may be defined as follows. Pick a new root x and consider the edge-weighted rooted trees $\mathfrak{T}'$ that are attached to the root in $\mathfrak{T}_1$ or $\mathfrak{T}_2$. Let $\mathfrak{w}_j$ for $j = 1, 2$ be the weight of the arc that attaches $\mathfrak{T}'$ to the root in $\mathfrak{T}_j$ (treating $\mathfrak{w}_j$ as 0 if such an arc does not exist). For each such $\mathfrak{T}'$, attach a copy of $\mathfrak{T}'$ to x through an arc with weight $\mathfrak{w}_1 + \mathfrak{w}_2$. For example,



This addition can be extended to arbitrary isomorphism types of finite edge-weighted directed rooted trees by setting $\mathfrak{T}_1 + \mathfrak{T}_2 := \mathrm{SF}(\mathfrak{T}_1) + \mathrm{SF}(\mathfrak{T}_2)$. Henceforth, we frequently drop the word "isomorphism type" (thus identifying a finite (edge-weighted) directed rooted tree with its isomorphism type) for the sake of simplicity.

4.1 Rooted trees under rigid procreation

Let $\Gamma = \Gamma_g$ be a finite functional graph such that Γ^* has rigid procreation (see Definition 2.1.7 (4)). Moreover, let $x \in \mathrm{V}(\Gamma)$ be g-periodic, and let $(\mathrm{proc}_k(x))_{k\geq 1}$ be the sequence of procreation numbers of x in Γ^* (which is independent of x due to rigid procreation). Proposition 2.1.8 states that the isomorphism type of the rooted tree $\mathrm{Tree}_\Gamma(x)$, which also does not depend on the choice of g-periodic vertex x, is entirely determined by this sequence of procreation numbers. We would like to understand explicitly how that isomorphism type can be derived from $(\mathrm{proc}_k(x))_{k\geq 1}$.

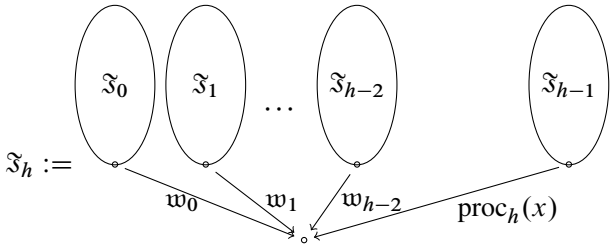
Since Γ is finite, so is $H := \mathrm{ht}(\mathrm{Tree}_\Gamma(x))$. We observe that x has g-transient children with at least $H - 1$ successor generations in Γ^*, but no such children with at

least H successor generations. This means that $\mathrm{proc}_h(x) > 1$ for $h = 1, 2, \ldots, H$, but $\mathrm{proc}_{H+1}(x) = 1$; the unique g-periodic child of x in Γ^* has infinitely many successor generations, thus providing a contribution of 1 to all procreation numbers. This allows us to read off H from the sequence $(\mathrm{proc}_k(x))_{k\geq 1}$ alone.

Now, Lemma 2.1.10 implies that for all g-transient vertices $y \in \mathrm{V}(\Gamma)$ with a fixed tree height $h \in \{0, 1, \ldots, H-1\}$ above them in Γ, the rooted tree isomorphism type $\mathrm{Tree}_\Gamma(y)$ is always the same. We recursively define a (not necessarily simplified) edge-weighted directed rooted tree $\mathfrak{T}_h = \mathfrak{T}_h((\mathrm{proc}_k(x))_{k\geq 1})$ such that the said isomorphism type is $\mathrm{Expand}(\mathfrak{T}_h)$. Clearly, the only choice for $\mathfrak{T}_0$ is a single vertex without arcs. If $h \in \{1, 2, \ldots, H-1\}$, then we define $\mathfrak{T}_h$ as follows. We fix a new root, and

- for $k = 0, 1, \ldots, h-2$, we attach a copy of $\mathfrak{T}_k$ to the new root with edge weight $\mathrm{proc}_{k+1}(x) - \mathrm{proc}_{k+2}(x) =: \mathfrak{w}_k$; and
- we attach a copy of $\mathfrak{T}_{h-1}$ to the new root with edge weight $\mathrm{proc}_h(x)$.

Here is a visual version of this definition.



This definition of $\mathfrak{T}_h$ does the job, because by the proof of Lemma 2.1.10, for each $k \in \{0, 1, \ldots, h-1\}$, the number of children z of y in Γ^* such that $\mathrm{Tree}_\Gamma(z)$ has height exactly $k-1$ (and hence is isomorphic to $\mathfrak{T}_{k-1}$ by induction) is equal to

$$\mathrm{proc}_{k+1}(y) - \mathrm{proc}_{k+2}(y) = \begin{cases} \mathrm{proc}_{k+1}(x) - \mathrm{proc}_{k+2}(x) = \mathfrak{w}_k, & \text{if } k < h-1, \\ \mathrm{proc}_h(x) - 0 = \mathrm{proc}_h(x), & \text{if } k = h-1. \end{cases}$$

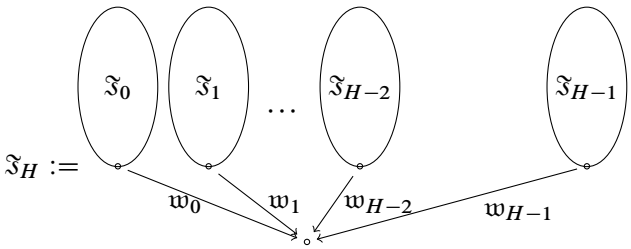
For our fixed periodic vertex x, the determination of $\mathrm{Tree}_\Gamma(x)$ is analogous, but one must take into account that x has a (unique) g-periodic child in Γ^*, which does not appear in $\mathrm{Tree}_\Gamma(x)$. This means that the weight with which $\mathfrak{T}_k$ for $k \in \{0, 1, \ldots, H-2\}$ is attached to the root of $\mathfrak{T}_H$ is

$$(\mathrm{proc}_{k+1}(x) - 1) - (\mathrm{proc}_{k+2}(x) - 1) = \mathfrak{w}_k,$$

whereas the weight with which $\mathfrak{T}_{H-1}$ is attached is

$$\mathrm{proc}_H(x) - 1 = \mathrm{proc}_H(x) - \mathrm{proc}_{H+1}(x) =: \mathfrak{w}_{H-1}.$$

In short, we obtain the following definition of $\mathfrak{T}_H$ such that $\mathrm{Expand}(\mathfrak{T}_H) \cong \mathrm{Tree}_\Gamma(x)$:



We can use similar ideas to describe, for each index d generalized cyclotomic mapping f of $\mathbb{F}_q$, the rooted trees above non-zero periodic vertices in Γ_{per}, the induced subgraph of Γ_f on the union of all periodic blocks C_i (in particular, we can obtain such a description for Γ_f as a whole in case $\bar{f}$ is a permutation). Let $i \in \{0, 1, \ldots, d-1\}$ be $\bar{f}$-periodic, with $\bar{f}$-cycle $(i_0, i_1, \ldots, i_{\ell-1})$, where $i_0 = i$. For $t \in \mathbb{Z}$, we set

$$i_t := i_{t \bmod \ell}.$$

Theorem 3.2.1 states that for fixed $t \in \mathbb{Z}$ and $h \in \mathbb{N}^+$, if $x, y \in C_{i_t}$ each have at least h successor generations in Γ^*_{per} (i.e., if $\min\{\mathrm{proc}_h^{(\Gamma^*_{\mathrm{per}})}(x), \mathrm{proc}_h^{(\Gamma^*_{\mathrm{per}})}(y)\} > 0$), then $\mathrm{proc}_h^{(\Gamma^*_{\mathrm{per}})}(x) = \mathrm{proc}_h^{(\Gamma^*_{\mathrm{per}})}(y)$. This allows us to set $\mathrm{proc}_{i_t,h} := \mathrm{proc}_h^{(\Gamma^*_{\mathrm{per}})}(x)$ for any $x \in C_{i_t}$ with at least h successor generations in Γ^*_{per} (such as an f-periodic x); this notation agrees with the one used in the proof of Theorem 3.2.1.

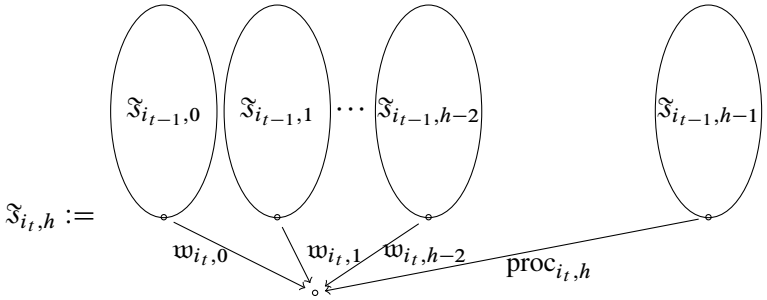
According to the comment before Theorem 3.2.1, for periodic $x \in C_{i_t}$, the isomorphism type of $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(x)$ only depends on i_t and the numbers $\mathrm{proc}_{i_{t'},h}$ for $h \geq 1$ and $t' \in \mathbb{Z}$ (i.e., it is independent of the choice of x). We describe how to read off this rooted tree from the data it depends on. We set

$$\mathcal{H}_{i_t} := \min\{h \in \mathbb{N}^+ : \mathrm{proc}_{i_t,h} = 1\} - 1,$$

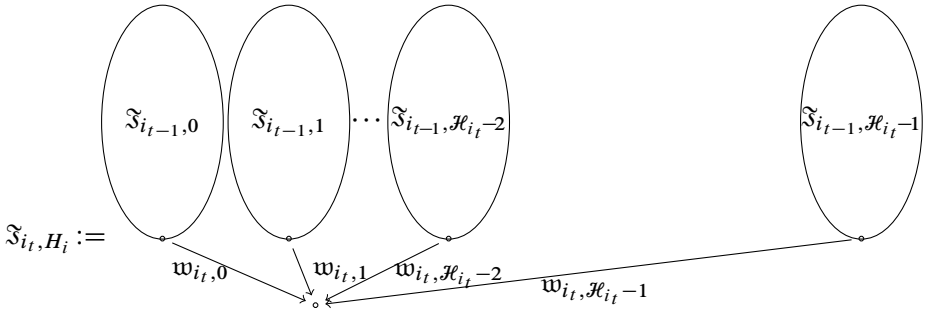
the common height of the rooted trees in Γ_{per} above periodic vertices in C_{i_t}. Moreover, as in Section 3.3, we let $H_i := \max\{\mathcal{H}_{i_t} : t = 0, 1, \ldots, \ell - 1\}$, the maximum such tree height along the cycle of i. Then, in generalization of what was stated above, for all vertices $x \in C_{i_t}$, the isomorphism type of $\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(x)$ only depends on i_t, the numbers $\mathrm{proc}_{i_{t'},h}$ and the $\mathfrak{h}$-value of x (see formula (3.1) in Section 3.3 for the definition of $\mathfrak{h}$). It should be noted that $\mathfrak{h}$ does not necessarily assume all of its possible values $0, 1, \ldots, H_i$ on each coset C_{i_t} (see Example 3.2.2), but this is not an issue for our construction.

We recursively define edge-weighted rooted trees $\mathfrak{T}_{i_t,h}$ such that the rooted tree in Γ_{per} above any $x \in C_{i_t}$ with $\mathfrak{h}(x) = h$ is isomorphic to $\mathrm{Expand}(\mathfrak{T}_{i_t,h})$, a property that is certainly satisfied whenever there are no $x \in C_{i_t}$ of that $\mathfrak{h}$-value. For $k \in \mathbb{N}_0$, we set $\mathfrak{w}_{i_t,k} := \mathrm{proc}_{i_t,k+1} - \mathrm{proc}_{i_t,k+2}$. We define $\mathfrak{T}_{i_t,0}$ to be the trivial rooted tree. If $h \in \{1, 2, \ldots, H_i - 1\}$ (we observe that vertices in C_{i_t} of such an $\mathfrak{h}$-value are

f-transient), then we set



Finally, the rooted tree above any vertex in C_{i_t} that is f-periodic (equivalently, which has $\mathfrak{h}$-value H_i) may be constructed as



4.2 An illustrative example

In this section, we follow the approach from Chapter 3 to derive the cyclic sequences of rooted tree isomorphism types that characterize the connected components of the functional graph of the following generalized cyclotomic mapping f of $\mathbb{F}_{2^8}$ of index $d = 5$:

$$f(x) = \begin{cases} 0, & \text{if } x = 0, \\ \omega^5 x^9, & \text{if } x \in C_0, \\ x^3, & \text{if } x \in C_1, \\ x^{17}, & \text{if } x \in C_2, \\ \omega^3 x^{34}, & \text{if } x \in C_3, \\ \omega^4 x^9, & \text{if } x \in C_4, \end{cases}$$

where ω is any fixed primitive element of $\mathbb{F}_{2^8}$ (the minimal polynomial of ω over $\mathbb{F}_2$ is not relevant here). These cyclic sequences were also derived in our introduction from a drawing of Γ_f (see the text passage between Definitions 1.4 and 1.5), but

the approach of Chapter 3 is usually more computationally efficient (see Chapter 5, especially Theorem 5.1.9).

We observe that if a generalized cyclotomic mapping of a finite field of known index is not given in the above cyclotomic form, but in polynomial form, then one must first convert it into cyclotomic form before one can apply our methods. An algorithm for doing so is [15, Algorithm 1].

Because $d = 5$, we have $s = (2^8 - 1)/5 = 51 = 3 \cdot 17$. We view each coset C_i as a copy of $\mathbb{Z}/51\mathbb{Z}$ via the bijection

$$\iota_i : \mathbb{Z}/51\mathbb{Z} \to C_i, \quad k \mapsto \omega^{i+5k}.$$

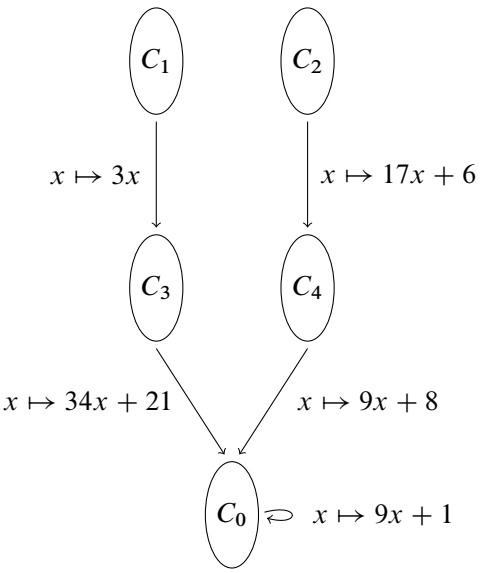
Let us work out what the monomial formulas for the values of f in the different cases become under this identification. For example, if $x \in C_4$, then $x = \omega^{4+5k}$ for some $k \in \mathbb{Z}$, and

$$f(x) = \omega^4 x^9 = \omega^{4+9\cdot 4+9\cdot 5k} = \omega^{0+5\cdot(9k+8)},$$

which shows that f maps C_4 to C_0 via the affine map

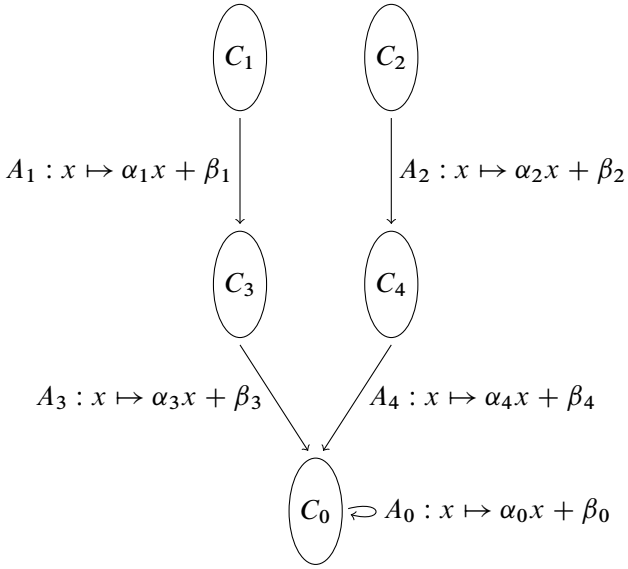
$$x \mapsto 9x + 8.$$

In total, we obtain the following picture describing the mapping behavior of f between the cosets C_i when viewing them as copies of $\mathbb{Z}/51\mathbb{Z}$.



Until further notice, we put the concrete function f from above aside and assume that, more generally, we have a finite field $\mathbb{F}_q$ with $5 \mid q - 1$ and an index 5 generalized

cyclotomic mapping f of $\mathbb{F}_q$ which maps as follows between the five cosets of C in $\mathbb{F}_q^*$, viewed as copies of $\mathbb{Z}/s\mathbb{Z}$ (where $s = (q-1)/5$).



This allows us to describe Γ_f in terms of those general coefficients α_i and β_i, which is more instructive; one can actually see the structure of formulas for relevant parameters, such as the moduli $\mathfrak{a}_{i,j}$ and right-hand sides $\mathfrak{b}_{i,j}$ of the spanning congruences of $\mathcal{P}_i$. Just as for our concrete generalized cyclotomic mapping from above, we assume that $\gcd(\alpha_0, s) = \gcd(\alpha_0^2, s) > 1$, which means that the rooted trees attached to periodic vertices in the induced subgraph of Γ_f on C_0 are of height $H_0 = 1$. We describe the arithmetic partitions $\mathcal{P}_i = \mathfrak{P}(x \equiv \mathfrak{b}_{i,j} \pmod{\mathfrak{a}_{i,j}} : j = 1, 2, \ldots, m_i)$ and the associated rooted tree isomorphism type $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ for each block $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ of $\mathcal{P}_i$ for $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$.

The partitions $\mathcal{P}_i$ and associated rooted trees are easily determined for $i = 1, 2, 3, 4$.

- For $i \in \{1, 2\}$, every vertex in C_i is a leaf in Γ_f, and so we may choose $\mathcal{P}_1 = \mathcal{P}_2 = \mathfrak{P}(\emptyset)$ (trivial partition with only one block). There is only one isomorphism type of rooted trees here, $\mathrm{Tree}_i(\mathcal{P}_i, \emptyset)$ (with $\emptyset$ representing an empty sequence of logical signs, not the positive logical sign), and it consists of a single vertex without edges.
- For $i \in \{3, 4\}$, since C_i is a transient coset (i.e., it does not lie on a cycle of cosets under f), the discussion in Section 3.3 shows that one can obtain $\mathcal{P}_i$ simply as the lift $\mathfrak{P}'(\mathcal{P}_{i-2}, A_{i-2})$. According to Lemma 2.2.2, that lift is of the form $\mathcal{P}_i = \mathfrak{P}(x \equiv \beta_{i-2} \pmod{\gcd(\alpha_{i-2}, s)})$. The significance of this single congruence is that it characterizes when $x \in C_i$ has at least one pre-image under f in C_{i-2}. We

note that if this is the case, then x has exactly $\gcd(\alpha_{i-2}, s)$ such pre-images, as they form a coset of the kernel of $z \mapsto \alpha_{i-2}z$ in $\mathbb{Z}/s\mathbb{Z}$. Hence, $\mathrm{Tree}_i(\mathcal{P}_i, (\neg))$ is a single vertex without arcs, and $\mathrm{Tree}_i(\mathcal{P}_i, (\emptyset))$ consists of a root with $\gcd(\alpha_{i-2}, s)$ vertices attached to it.

In our discussion for $\mathcal{P}_0$, rather than specify the rooted tree $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ associated with a block $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ of $\mathcal{P}_0$ itself, we specify a (not necessarily simplified isomorphism type of) finite edge-weighted directed rooted tree(s) $\mathfrak{T} = \mathfrak{T}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ such that $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)}) = \mathrm{Expand}(\mathfrak{T})$. But first, let us determine $\mathcal{P}_0$ itself. We recall that $H_0 = 1$ by assumption. According to the general definition of $\mathcal{P}_i$ for periodic i, which is just before Proposition 3.3.5, we have

$$\mathcal{P}_0 = \mathcal{Q}_{0,1} = \mathcal{P}_{0,1} \wedge \mathcal{U}_0.$$

Moreover, noting that $i = 0$ lies on a cycle of $\bar{f}$ of length 1 (so that $i_t = 0$ for all $t \in \mathbb{Z}$ in the notation of Section 3.3), we conclude that

$$\mathcal{P}_{0,1} = \lambda_0^0(\mathcal{R}_0) \wedge \lambda_0^1(\mathcal{R}_0) = \mathcal{R}_0 \wedge \lambda(\mathcal{R}_0, A_0).$$

Now, $\mathcal{R}_0$ is obtained as the infimum of the $\mathfrak{P}'$-lifts of $\mathcal{P}_3$ and $\mathcal{P}_4$ to C_0. Using the notation (n, m) in place of $\gcd(n, m)$ for simplicity, we conclude that

$$\mathcal{R}_0 = \mathfrak{P}'(\mathcal{P}_3, A_3) \wedge \mathfrak{P}'(\mathcal{P}_4, A_4) = \mathfrak{P}\begin{pmatrix} x \equiv \alpha_3\beta_1 + \beta_3 \pmod{(\alpha_3(\alpha_1, s), s)} \\ x \equiv \beta_3 \pmod{(\alpha_3, s)} \\ x \equiv \alpha_4\beta_2 + \beta_4 \pmod{(\alpha_4(\alpha_2, s), s)} \\ x \equiv \beta_4 \pmod{(\alpha_4, s)} \end{pmatrix}$$

$$= \mathfrak{P}\begin{pmatrix} x \equiv \alpha_3\beta_1 + \beta_3 \pmod{(\alpha_1, s)(\alpha_3, \frac{s}{(\alpha_1, s)})} \\ x \equiv \beta_3 \pmod{(\alpha_3, s)} \\ x \equiv \alpha_4\beta_2 + \beta_4 \pmod{(\alpha_2, s)(\alpha_4, \frac{s}{(\alpha_2, s)})} \\ x \equiv \beta_4 \pmod{(\alpha_4, s)} \end{pmatrix}, \tag{4.1}$$

and thus

$$\lambda(\mathcal{R}_0, A_0) = \mathfrak{P}\begin{pmatrix} x \equiv \alpha_0\alpha_3\beta_1 + \alpha_0\beta_3 + \beta_0 \pmod{(\alpha_0(\alpha_1, s)(\alpha_3, \frac{s}{(\alpha_1, s)}), s)} \\ x \equiv \alpha_0\beta_3 + \beta_0 \pmod{(\alpha_0(\alpha_3, s), s)} \\ x \equiv \alpha_0\alpha_4\beta_2 + \alpha_0\beta_4 + \beta_0 \pmod{(\alpha_0(\alpha_2, s)(\alpha_4, \frac{s}{(\alpha_2, s)}), s)} \\ x \equiv \alpha_0\beta_4 + \beta_0 \pmod{(\alpha_0(\alpha_4, s), s)} \end{pmatrix}$$

$$= \mathfrak{P}\begin{pmatrix} x \equiv \alpha_0\alpha_3\beta_1 + \alpha_0\beta_3 + \beta_0 \pmod{(\alpha_1, s)(\alpha_0(\alpha_3, \frac{s}{(\alpha_1, s)}), \frac{s}{(\alpha_1, s)})} \\ x \equiv \alpha_0\beta_3 + \beta_0 \pmod{(\alpha_0(\alpha_3, s), s)} \\ x \equiv \alpha_0\alpha_4\beta_2 + \alpha_0\beta_4 + \beta_0 \pmod{(\alpha_2, s)(\alpha_0(\alpha_4, \frac{s}{(\alpha_2, s)}), \frac{s}{(\alpha_2, s)})} \\ x \equiv \alpha_0\beta_4 + \beta_0 \pmod{(\alpha_0(\alpha_4, s), s)} \end{pmatrix}.$$

Moreover, by formula (3.3) and the definition of $\mathcal{U}_i$ just after it, we have

$$\mathcal{U}_0 = \mathfrak{P}(\theta_{0,1}) = \mathfrak{P}(x \equiv \beta_0 \pmod{(\alpha_0, s)}).$$

It follows that

$$\mathcal{P}_0 = \mathfrak{P}\left(\begin{array}{l} x \equiv \alpha_3\beta_1 + \beta_3 \ (\mathrm{mod}\ (\alpha_1, s)(\alpha_3, \frac{s}{(\alpha_1,s)})) \\ x \equiv \beta_3 \ (\mathrm{mod}\ (\alpha_3, s)) \\ x \equiv \alpha_4\beta_2 + \beta_4 \ (\mathrm{mod}\ (\alpha_2, s)(\alpha_4, \frac{s}{(\alpha_2,s)})) \\ x \equiv \beta_4 \ (\mathrm{mod}\ (\alpha_4, s)) \\ x \equiv \alpha_0\alpha_3\beta_1 + \alpha_0\beta_3 + \beta_0 \ (\mathrm{mod}\ (\alpha_1, s)(\alpha_0(\alpha_3, \frac{s}{(\alpha_1,s)}), \frac{s}{(\alpha_1,s)})) \\ x \equiv \alpha_0\beta_3 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_3, s), s)) \\ x \equiv \alpha_0\alpha_4\beta_2 + \alpha_0\beta_4 + \beta_0 \ (\mathrm{mod}\ (\alpha_2, s)(\alpha_0(\alpha_4, \frac{s}{(\alpha_2,s)}), \frac{s}{(\alpha_2,s)})) \\ x \equiv \alpha_0\beta_4 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_4, s), s)) \\ x \equiv \beta_0 \ (\mathrm{mod}\ (\alpha_0, s)) \end{array}\right). \tag{4.2}$$

Now we turn to the determination of the rooted trees above vertices in any given block B of $\mathcal{P}_0$. More specifically, we have

$$B = \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)}),$$

where $\vec{\nu}^{(\mathcal{P}_0)} = (\nu_1, \ldots, \nu_9) \in \{\emptyset, \neg\}^9$ is a tuple of logical signs for the nine spanning congruences of $\mathcal{P}_0$. It is helpful to split $\vec{\nu}^{(\mathcal{P}_0)}$ into segments; namely, in the notation of Proposition 3.3.5, we write $\vec{\nu}^{(\mathcal{P}_0)} = \vec{o'_0} \diamond \vec{o'_1} \diamond \vec{\xi}$, where

- $\vec{o'_0} = (\nu_1, \nu_2, \nu_3, \nu_4)$ controls in which block $\mathcal{B}(\mathcal{R}_0, \vec{o'_0})$ of $\mathcal{R}_0$ the $\mathcal{P}_0$-block $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ is contained. By Proposition 3.3.3, knowing the logical signs in $\vec{o'_0}$ is enough to understand, uniformly for all $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$, the contribution

  $$\mathrm{Tree}_{\Gamma_f}(x, C_3 \cup C_4) = \mathrm{Tree}_0(\mathcal{R}_0, C_3 \cup C_4, \vec{o'_0})$$

  to $\mathrm{Tree}_{\Gamma_f}(x)$ that comes from those pre-images of x that lie in $C_3 \cup C_4$ (the union of all transient cosets that map to C_0).
- $\vec{o'_1} = (\nu_5, \nu_6, \nu_7, \nu_8)$ controls in which block $\mathcal{B}(\mathcal{S}_{0,1}, \vec{o'_1})$ of $\mathcal{S}_{0,1} = \lambda(\mathcal{R}_0, A_0)$ the $\mathcal{P}_0$-block $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ is contained. By Proposition 3.3.4, knowing the logical signs in $\vec{o'_1}$ is enough to understand, uniformly for all $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ *of* $\mathfrak{h}$*-value* 1, i.e., which are f-periodic (or, equivalently here, which are non-leaves in Γ_{per}), the contribution $\mathrm{Tree}_{\Gamma_f}(x, C_0) = \mathrm{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o'_1})$ to $\mathrm{Tree}_{\Gamma_f}(x)$ that comes from those pre-images of x that lie in C_0 (the unique periodic coset that maps to C_0). We note that if $x \in C_0$ has $\mathfrak{h}$-value 0, i.e., if x is f-transient (or, equivalently here, if x is a leaf in Γ_{per}), then $\mathrm{Tree}_{\Gamma_f}(x, C_0)$ is trivial, because x has no f-transient pre-images in C_0.
- $\vec{\xi} = (\nu_9)$ controls the $\mathfrak{h}$-value of the vertices in $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$; if $\nu_9 = \neg$, then all of those vertices are leaves in Γ_{per} (i.e., their $\mathfrak{h}$-value is 0), otherwise they all are periodic vertices (i.e., their $\mathfrak{h}$-value is $1 = H_0$).

Let us be more specific about these different contributions to $\mathrm{Tree}_{\Gamma_f}(x)$ for $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$. We recall that by definition,

$$\mathcal{R}_0 = \mathcal{P}_3' \wedge \mathcal{P}_4' = \mathfrak{P}'(\mathcal{P}_3, A_3) \wedge \mathfrak{P}'(\mathcal{P}_4, A_4),$$

and note that $\vec{o}_0'$ can be written as the concatenation $\vec{\nu}^{(\mathcal{P}_3')} \diamond \vec{\nu}^{(\mathcal{P}_4')}$, with $\vec{\nu}^{(\mathcal{P}_3')} = (\nu_1, \nu_2)$, respectively,

$$\vec{\nu}^{(\mathcal{P}_4')} = (\nu_3, \nu_4),$$

controlling the containment of $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ in $\mathcal{P}_3'$-blocks, respectively, in $\mathcal{P}_4'$-blocks. For $i \in \{3, 4\}$, knowing the logical signs in $\vec{\nu}^{(\mathcal{P}_i')}$ is enough to understand, uniformly for all $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$, the contribution

$$\mathrm{Tree}_{\Gamma_f}(x, C_i) = \mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$$

to $\mathrm{Tree}_{\Gamma_f}(x)$ that comes from those pre-images of x that lie in C_i.

Of course, for each given $x \in C_0$, we have

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}(x, C_0 \cup C_3 \cup C_4) = \sum_{i \in \{0,3,4\}} \mathrm{Tree}_{\Gamma_f}(x, C_i).$$

In view of what was said above about these three different contributions to $\mathrm{Tree}_{\Gamma_f}(x)$, we have the following formulas (which can also be derived from Propositions 3.3.3 (2) and 3.3.5 as well as the last formula in Proposition 3.3.4):

$$\begin{aligned} &\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)}) \\ &= \begin{cases} \sum_{i=3}^4 \mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')}), & \text{if } \nu_9 = \neg, \\ \sum_{i=3}^4 \mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')}) + \mathrm{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o}_1'), & \text{if } \nu_9 = \emptyset. \end{cases} \end{aligned} \tag{4.3}$$

In particular, the logical signs $\nu_5, \nu_6, \nu_7, \nu_8$ in $\vec{o}_1'$ are irrelevant for the value of $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ if $\nu_9 = \neg$.

Formula (4.3) allows us to split the task of determining $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ into subtasks. First, we determine $\mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$ for $i = 3, 4$, which can be done uniformly. We note that

$$\mathcal{P}_i' = \mathfrak{P}\begin{pmatrix} x \equiv \alpha_i\beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)} \\ x \equiv \beta_i \pmod{(\alpha_i, s)} \end{pmatrix}.$$

The two entries of $\vec{\nu}^{(\mathcal{P}_i')} = (\nu_{2i-5}, \nu_{2i-4})$ are logical signs for those two congruences, and we need to distinguish cases according to their truth values. We could just work out $\mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$ in each case "mechanically" following Proposition 3.3.3 (1), using the formula for $\sigma_{\mathcal{P}_i, A_i}(\vec{\nu}^{(\mathcal{P}_i)}, \vec{\nu}^{(\mathcal{P}_i')})$ from Lemma 2.2.2. However, it is more instructive to derive them with direct arguments (inclined readers may still follow the formulaic approach themselves and compare).

- If $\nu_{2i-4} = \neg$, i.e., if $x \not\equiv \beta_i \pmod{(\alpha_i, s)}$, then x simply has no pre-images in C_i, whence $\mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$ is a single vertex without arcs (the value of ν_{2i-5} is irrelevant here).
- If $\nu_{2i-4} = \emptyset$, i.e., if $x \equiv \beta_i \pmod{(\alpha_i, s)}$, then x has exactly (α_i, s) children in C_i (which form a coset of the kernel of $z \mapsto \alpha_i z$). We need to determine the distribution of those children over the two blocks of

$$\mathcal{P}_i = \mathfrak{P}(x \equiv \beta_{i-2} \pmod{(\alpha_{i-2}, s)}),$$

and that distribution is controlled by the truth value ν_{2i-5} of

$$x \equiv \alpha_i \beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)}. \tag{4.4}$$

Indeed, following the proof of Lemma 2.2.2, the pre-images y of x in C_i that satisfy

$$y \equiv \beta_{i-2} \pmod{(\alpha_{i-2}, s)}$$

(we note that they are exactly those pre-images of x in C_i which are *not* leaves in Γ_f) are characterized by the system of congruences

$$\begin{aligned} y &\equiv \beta_{i-2} \pmod{(\alpha_{i-2}, s)} \\ \alpha_i y + \beta_i &\equiv x \pmod{s}, \end{aligned}$$

which is (according to the proof of Lemma 2.2.2) consistent if and only if congruence (4.4) holds, in which case the system is equivalent to a single congruence modulo

$$\mathrm{lcm}\Big((\alpha_{i-2}, s), \frac{s}{(\alpha_i, s)}\Big).$$

Hence, if $\nu_{2i-5} = \emptyset$, i.e., if congruence (4.4) holds, then x has exactly

$$\frac{s}{\mathrm{lcm}\big((\alpha_{i-2}, s), \frac{s}{(\alpha_i, s)}\big)} = \Big(\frac{s}{(\alpha_{i-2}, s)}, (\alpha_i, s)\Big)$$

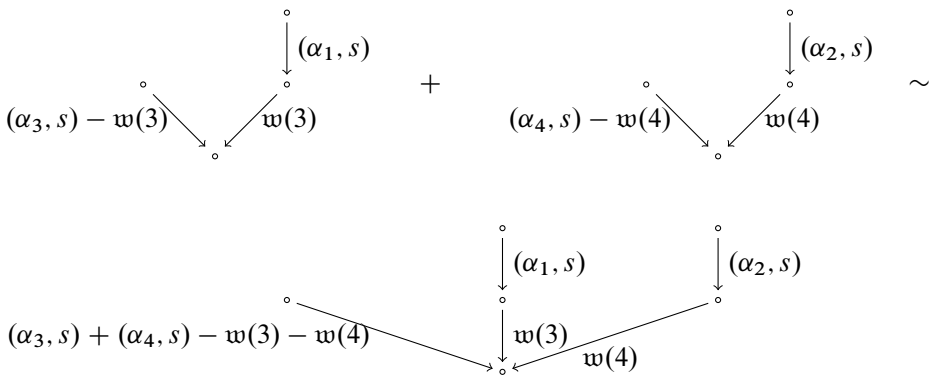
pre-images $y \in C_i$ with $y \equiv \beta_{i-2} \pmod{(\alpha_{i-2}, s)}$, which are exactly those pre-images that lie in $\mathcal{B}(\mathcal{P}_i, (\emptyset))$. Otherwise, all pre-images of x in C_i are incongruent to β_{i-2} modulo (α_{i-2}, s) and thus lie in $\mathcal{B}(\mathcal{P}_i, (\neg))$. In view of the known value of $\mathrm{Tree}_i(\mathcal{P}_i, (\nu))$ in terms of $\nu \in \{\emptyset, \neg\}$, we find that $\mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$ is the expanded version of the (not necessarily simplified) edge-weighted directed rooted tree specified in Table 4.1.

This settles the first two summands of $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ in each of the two cases in formula (4.3). If $\nu_9 = \neg$ (i.e., if the $\mathcal{P}_0$-block in question consists of f-transient points), then these are all the summands in the formula, and one can obtain (an edge-weighted version of) $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ simply by adding (the edge-weighted versions

block of $\mathcal{P}'_i = \mathfrak{B}'(\mathcal{P}_i, A_i)$	associated $\mathrm{Tree}_{\Gamma_f}(x, C_i)$
$\nu(x \equiv \alpha_i \beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)})$ $x \not\equiv \beta_i \pmod{(\alpha_i, s)}$ $\nu \in \{\emptyset, \neg\}$	
$x \not\equiv \alpha_i \beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)}$ $x \equiv \beta_i \pmod{(\alpha_i, s)}$	
$x \equiv \alpha_i \beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)}$ $x \equiv \beta_i \pmod{(\alpha_i, s)}$	

Table 4.1. Rooted trees using only pre-images in the transient pre-image coset C_i with $i \in \{3, 4\}$.

of) $\mathrm{Tree}_i(\mathcal{P}'_i, \vec{\nu}^{(\mathcal{P}'_i)})$ for $i \in \{3, 4\}$, read off from Table 4.1. For example, if $\nu_j = \emptyset$ for $j = 1, 2, \ldots, 8$ but $\nu_9 = \neg$, then (an edge-weighted version of) $\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ is as follows, setting $\mathfrak{w}(i) := (\frac{s}{(\alpha_i - 2, s)}, (\alpha_i, s))$ for $i = 3, 4$:



On the other hand, if $\nu_9 = \emptyset$ (so that all vertices in $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ are f-periodic), then we also need to compute $\mathrm{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o'_1})$, the third summand in formula (4.3), which expresses the contribution coming from f-transient pre-images in C_0. Following Proposition 3.3.4, this can be done by studying the distribution of pre-images of any given point $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ over certain blocks of the partition $\mathcal{Q}_{0,0} = \mathcal{R}_0 \wedge \mathfrak{P}(x \equiv \beta_0 \pmod{(\alpha_0, s)})$. More specifically, we note that each f-transient pre-image of x is contained in a block of $\mathcal{Q}_{0,0}$ of the form $\mathcal{B}(\mathcal{Q}_{0,0}, \vec{o_0} \diamond (\neg))$ for some $\vec{o_0} \in \{\emptyset, \neg\}^4$ (and we also observe that each block of $\mathcal{Q}_{0,0}$ of this form consists entirely of f-transient points, due to the last logical sign being $\neg$). Being able to

count the number of pre-images of x in each such block of $\mathcal{Q}_{0,0}$ is enough to understand $\mathrm{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o_1'})$, because $\mathcal{B}(\mathcal{Q}_{0,0}, \vec{o_0} \diamond (\neg)) \subseteq \mathcal{B}(\mathcal{R}_0, \vec{o_0})$ and we already understand the rooted trees above f-transient vertices in a given block $\mathcal{B}(\mathcal{R}_0, \vec{o_0'}) = \mathcal{B}(\mathcal{R}_0, \vec{\nu}^{(\mathcal{P}_3')} \diamond \vec{\nu}^{(\mathcal{P}_4')})$ of $\mathcal{R}_0$.

Now, let us observe that the distribution of the pre-images of any f-periodic point $x \in C_0$ over the blocks of $\mathcal{Q}_{0,0}$ is controlled by the values of ν_j for $j \in \{5, 6, 7, 8\}$, i.e., by the block $\mathcal{B}(\mathcal{S}_{0,1}, \vec{o_1'})$ of $\mathcal{S}_{0,1}$ in which x is contained. This is because $x \in \mathcal{B}(\mathcal{T}_{0,1}, \vec{o_1'} \diamond (\emptyset))$, where $\mathcal{T}_{0,1} = \mathcal{S}_{0,1} \wedge \mathcal{U}_0 = \mathfrak{P}'(\mathcal{Q}_{0,0}, A_0)$, a partition which does indeed control the distribution of pre-images of x over the blocks of $\mathcal{Q}_{0,0}$ according to Lemma 2.2.2. Applying this lemma here leads to the formula for $\mathrm{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o_1'})$ from Proposition 3.3.4.

For example, the logical sign tuple $\vec{o_0} = (\neg, \neg, \emptyset, \emptyset)$ corresponds to the block $B := \mathcal{B}(\mathcal{R}_0, \vec{o_0})$ of $\mathcal{R}_0$. If we wish to count how many transient pre-images a vertex $x \in C_i$ with $x \equiv \alpha_0 \pmod{(\alpha_0, s)}$ stemming from, say, the block $B' := \mathcal{B}(\mathcal{S}_{0,1}, \vec{o_1'})$ of $\mathcal{S}_{0,1}$ with $\vec{o_1'} = (\emptyset, \neg, \emptyset, \emptyset)$ has, then we need to compute

$$\sigma_{\mathcal{Q}_{0,0}, A_0}(\vec{o_0} \diamond (\neg), \vec{o_1'} \diamond (\emptyset)),$$

which we do now to illustrate the method. To avoid confusion among readers, we note that the above expression does not perfectly match the notation used in Lemma 2.2.2. Indeed, here we use spanning congruence sequences of length 5 both for $\mathcal{Q}_{0,0} = \mathcal{R}_0 \wedge \mathcal{T}_{0,0} = \mathcal{R}_0 \wedge \mathcal{U}_0$ and for $\mathcal{T}_{0,1} = \mathfrak{P}'(\mathcal{Q}_{0,0}, A_0)$. However, in Lemma 2.2.2, it is assumed that we use the "standard format" of the spanning congruence sequence for $\mathfrak{P}'(\mathcal{Q}_{0,0}, A_0)$, which contains one congruence more than the sequence for $\mathcal{Q}_{0,0}$. This discrepancy occurs because we write $\mathcal{T}_{0,1}$ as $\lambda(\mathcal{R}_0, A_0) \wedge \mathcal{U}_0$ – in the "standard format", it would instead be

$$\begin{aligned}&\lambda(\mathcal{Q}_{0,0}, A_0) \wedge \mathfrak{P}(x \equiv \beta_0 \pmod{(\alpha_0, s)})\\ &\quad = \lambda(\mathcal{R}_0, A_0) \wedge \mathfrak{P}(x \equiv \beta_0(1+\alpha_0) \pmod{(\alpha_0^2, s)}) \wedge \mathcal{U}_0,\end{aligned}$$

but we can omit the congruence $x \equiv \beta_0(1+\alpha_0) \pmod{(\alpha_0^2, s)}$, which is $\theta_{0,2}(x)$ in the notation of Section 3.3, because (using that $H_0 = 1$) it is equivalent to $x \equiv \beta_0 \pmod{(\alpha_0, s)}$, the unique spanning congruence $\theta_{0,1}(x)$ of $\mathcal{U}_0$. In order to apply Lemma 2.2.2, we put $\mathcal{T}_{0,1}$ into the less concise standard format, which requires us to replace the logical sign sequence $\vec{o_1'} \diamond (\emptyset) = (\emptyset, \neg, \emptyset, \emptyset, \emptyset)$ for the block of $\mathcal{T}_{0,1}$ by $\vec{\nu'} := \vec{o_1'} \diamond (\emptyset, \emptyset) = (\emptyset, \neg, \emptyset, \emptyset, \emptyset, \emptyset)$ (i.e., we double the $\emptyset$ at the end), the j-th entry of which we denote by ν_j'. The logical sign sequence for the block of $\mathcal{Q}_{0,0}$ remains $\vec{\nu} := \vec{o_0} \diamond (\neg) = (\neg, \neg, \emptyset, \emptyset, \neg)$. Our goal now is to compute $\sigma_{\mathcal{Q}_{0,0}, A_0}(\vec{\nu}, \vec{\nu'})$ strictly following Lemma 2.2.2. For $j = 1, \ldots, 5$, we denote by $\bar{\mathfrak{a}}_j$, respectively, $\bar{\mathfrak{b}}_j$, the modulus, respectively, right-hand side, of the j-th spanning congruence of $\mathcal{Q}_{0,0} = \mathcal{R}_0 \wedge \mathcal{U}_0$. That is, for $j \in \{1, 2, 3, 4\}$, the congruence $x \equiv \bar{\mathfrak{b}}_j \pmod{\bar{\mathfrak{a}}_j}$ is the

j-th displayed congruence in the formula for $\mathcal{P}_0$, (4.2). Moreover, $\bar{\mathfrak{a}}_5 = (\alpha_0, s)$ and $\bar{\mathfrak{b}}_5 = \beta_0$. Using the notation from Lemma 2.2.2, we observe that

- $J_-(\vec{\nu}) = \{1, 2, 5\}$ (the set of indices $j \in \{1, \dots, 5\}$ such that the j-th entry of $\vec{o_0} \diamond (\neg)$ is $\neg$);
- $J_+(\vec{\nu}) = \{1, 2, 3, 4, 5\} \setminus J_-(\vec{\nu}) = \{3, 4\}$;
- $J_-(\vec{\nu'}) = \{2\}$;
- for $J \subseteq J_-(\vec{\nu}) = \{1, 2, 5\}$, the condition $E(\vec{\nu}, J)$ demands: "For all $j_1, j_2 \in \{3, 4\} \cup J$: $\gcd(\bar{\mathfrak{a}}_{j_1}, \bar{\mathfrak{a}}_{j_2}) \mid \bar{\mathfrak{b}}_{j_1} - \bar{\mathfrak{b}}_{j_2}$".

According to Lemma 2.2.2, we have

$$\sigma_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}) = \sum_{J \subseteq J_-(\vec{\nu})} (-1)^{|J|} \kappa_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}, J),$$

where

$$\kappa_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}, J) = \delta_{\nu'_6=\emptyset} \cdot \delta_{E(\vec{\nu},J)} \cdot \delta_{(J_+(\vec{\nu})\cup J)\cap J_-(\vec{\nu'})=\emptyset} \cdot \frac{s}{\operatorname{lcm}(\frac{s}{\gcd(\alpha_0,s)}, \bar{\mathfrak{a}}_{0,j} : j \in J_+(\vec{\nu}) \cup J)}.$$

In this formula, the first Kronecker delta checks whether the last entry of $\vec{\nu'}$ is $\emptyset$, which is the case. The third Kronecker delta is 1 if and only if $2 \notin J$, which leaves the four possibilities $\emptyset, \{1\}, \{5\}, \{1, 5\}$ for $J \subseteq J_-(\vec{\nu}) = \{1, 2, 5\}$ for which $\kappa_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}, J)$ is potentially non-zero. We conclude that

$$\begin{aligned}\sigma_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}) = {} & \delta_{E(\vec{\nu},\emptyset)} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \bar{\mathfrak{a}}_3, \bar{\mathfrak{a}}_4)} - \delta_{E(\vec{\nu},\{1\})} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \bar{\mathfrak{a}}_1, \bar{\mathfrak{a}}_3, \bar{\mathfrak{a}}_4)} \\ & - \delta_{E(\vec{\nu},\{5\})} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \bar{\mathfrak{a}}_3, \bar{\mathfrak{a}}_4, \bar{\mathfrak{a}}_5)} \\ & + \delta_{E(\vec{\nu},\{1,5\})} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \bar{\mathfrak{a}}_1, \bar{\mathfrak{a}}_3, \bar{\mathfrak{a}}_4, \bar{\mathfrak{a}}_5)}\end{aligned}$$

is the number of f-transient children in $B = \mathcal{B}(\mathcal{R}_0, \vec{o_0})$ of each given $x \in B' = \mathcal{B}(\mathcal{S}_{0,1}, \vec{o'_1})$. Each of these children provides a copy of

$$\operatorname{Tree}_0^{(0)}(\mathcal{P}_{0,0}, \vec{o_0}) = \operatorname{Tree}_0(\mathcal{R}_0, C_3 \cup C_4, \vec{o_0})$$

that is attached to the root of $\operatorname{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o'_1})$. If we carry this computation out for fixed $\vec{o'_1}$ and all possible values of $\vec{o_0} \in \{\emptyset, \neg\}^4$, then we obtain a "complete picture" of $\operatorname{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o'_1})$.

It is time to particularize the gained explicit understanding of the rooted trees back to the concrete example we started from. First, we deal with the rooted trees

$\mathrm{Tree}_{\Gamma_f}(x, C_3 \cup C_4)$ in terms of the blocks of $\mathcal{R}_0$. By substituting $s = 51$, $\alpha_0 = 9$, $\alpha_1 = 3$, $\alpha_2 = 17$, $\alpha_3 = 34$, $\alpha_4 = 9$, $\beta_0 = 1$, $\beta_1 = 0$, $\beta_2 = 6$, $\beta_3 = 21$, $\beta_4 = 8$ into formula (4.1), we get

$$\mathcal{R}_0 = \mathfrak{P}\begin{pmatrix} x \equiv 21 \pmod{51} \\ x \equiv 4 \pmod{17} \\ x \equiv 11 \pmod{51} \\ x \equiv 2 \pmod{3} \end{pmatrix}.$$

There are dependencies between these congruences. For example, the first implies the second as well as the negations of the third and fourth. Table 4.2 lists all $\vec{o_0'} \in \{\emptyset, \neg\}^4$ such that $\mathcal{B}(\mathcal{R}_0, \vec{o_0'})$ is nonempty, along with a description of the set $\mathcal{B}(\mathcal{R}_0, \vec{o_0'})$ and the (simplified edge-weighted form of the) corresponding rooted tree $\mathrm{Tree}_0(\mathcal{R}_0, C_3 \cup C_4, \vec{o_0'})$ above each point in $\mathcal{B}(\mathcal{R}_0, \vec{o_0'})$, obtained by adding the (edge-weighted forms of the) rooted trees $\mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{v_i'})$ for $i = 3, 4$ read off from Table 4.1.

Now we turn to the description of $\mathrm{Tree}_{\Gamma_f}(x, C_0)$ for f-periodic $x \in C_0$ in terms of the $\mathcal{S}_{0,1}$-block $\mathcal{B}(\mathcal{S}_{0,1}, \vec{o_1'})$ in which x lies. First, we substitute our concrete values of s and the α_i and β_i into the four spanning congruences for $\mathcal{S}_{0,1}$ to get that

$$\mathcal{S}_{0,1} = \mathfrak{P}\begin{pmatrix} x \equiv 37 \pmod{51} \\ x \equiv 37 \pmod{51} \\ x \equiv 49 \pmod{51} \\ x \equiv 1 \pmod{3} \end{pmatrix}.$$

It is not necessary to strictly follow the computations described in Lemma 2.2.2 (as outlined above) to gain a complete understanding of the trees here – we give a conceptual argument instead.

We note that by Lemma 2.1.14, the points in $\mathbb{Z}/51\mathbb{Z}$ that are periodic under $A_0 : x \mapsto 9x + 1$ are just those that are congruent to 1 modulo 3 (the unique fixed point of A_0 modulo $3 = \gcd(\alpha_0^L, s)$, where L is as in Lemma 2.1.14). Hence, the last spanning congruence of $\mathcal{S}_{0,1}$ is always true for those x we are considering. We observe that it is just a coincidence that the spanning sequence for $\mathcal{S}_{0,1}$ contains the characterizing congruence for periodic vertices – in general, this information needs to be added "externally" if one wants to control it via the partition blocks, using the partition $\mathcal{T}_{0,1} = \mathcal{S}_{0,1} \wedge \mathcal{U}_0$ instead.

Since the children of x in C_0 form a coset of the kernel $17\mathbb{Z}/51\mathbb{Z}$ of $z \mapsto 9z$, it follows that x has precisely three pre-images in C_0, one in each congruence class modulo 3. But the pre-image y of x in C_0 with $y \equiv 1 \pmod 3$ is f-periodic and hence does not occur in $\mathrm{Tree}_{\Gamma_f}(x, C_0)$. We also note the following.

- For $x = 37$, the remaining two, f-transient pre-images are 21 and 38.
- For $x = 49$ the f-transient pre-images are 11 and 45.

$\vec{o_0'}$	$B := \mathcal{B}(\mathcal{R}_0, \vec{o_0'})$	$\mathrm{Tree}_{\Gamma_f}(x, C_3 \cup C_4)$ for $x \in B$
$\begin{pmatrix} \neg \\ \neg \\ \neg \\ \neg \end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \not\equiv 2 \pmod 3\} \setminus \{4, 21\}$	∘
$\begin{pmatrix} \neg \\ \emptyset \\ \neg \\ \neg \end{pmatrix}$	$\{4\}$	∘ ↓ 17 ∘
$\begin{pmatrix} \emptyset \\ \emptyset \\ \neg \\ \neg \end{pmatrix}$	$\{21\}$	∘ ↓ 3 ∘ ↓ 17 ∘
$\begin{pmatrix} \neg \\ \neg \\ \neg \\ \emptyset \end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \equiv 2 \pmod 3\} \setminus \{11, 38\}$	∘ ↓ 3 ∘
$\begin{pmatrix} \neg \\ \neg \\ \emptyset \\ \emptyset \end{pmatrix}$	$\{11\}$	∘ ↓ 17 ∘ ↓ 3 ∘
$\begin{pmatrix} \neg \\ \emptyset \\ \neg \\ \emptyset \end{pmatrix}$	$\{38\}$	∘ ↓ 20 ∘

Table 4.2. Rooted trees using only pre-images in transient pre-image cosets.

- For all other $x \equiv 1 \pmod 3$, there is one f-transient pre-image each in the two "generic" blocks $\mathcal{B}(\mathcal{R}_0, (\neg, \neg, \neg, \neg))$ and $\mathcal{B}(\mathcal{R}_0, (\neg, \neg, \neg, \emptyset))$ of $\mathcal{R}_0$; this is because all other blocks in Table 4.2 except $\{4\}$ have already been "used up", and $4 \equiv 1 \pmod 3$.

From Table 4.2, we can read off $\mathrm{Tree}_{\Gamma_f}(y, C_3 \cup C_4) = \mathrm{Tree}_{\Gamma_f}(y)$ for each of the two f-transient pre-images y of x in C_0, thus obtaining the shape of $\mathrm{Tree}_{\Gamma_f}(x, C_0)$ specified in Table 4.3.

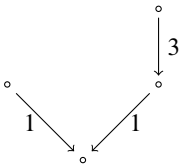
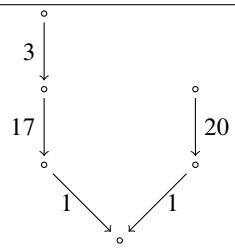
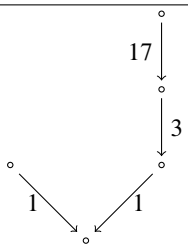
$\vec{o}'_1$	$B' := \mathcal{B}(\mathcal{S}_{0,1}, \vec{o}'_1)$	$\mathrm{Tree}_{\Gamma_f}(x, C_0)$ for periodic $x \in B'$
$\begin{pmatrix} \neg \\ \neg \\ \neg \\ \neg \end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \not\equiv 1 \pmod 3\}$	n/a (no periodic x in this block)
$\begin{pmatrix} \neg \\ \neg \\ \neg \\ \emptyset \end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \equiv 1 \pmod 3\} \setminus \{37, 49\}$	
$\begin{pmatrix} \emptyset \\ \emptyset \\ \neg \\ \emptyset \end{pmatrix}$	$\{37\}$	
$\begin{pmatrix} \neg \\ \neg \\ \emptyset \\ \emptyset \end{pmatrix}$	$\{49\}$	

Table 4.3. Rooted trees using only pre-images in C_0.

Finally, our formula (4.3) leads us to a tabular list of $\mathrm{Tree}_{\Gamma_f}(x)$, which we specify in Table 4.4, where we also introduce the notation $\mathfrak{T}_n$ for $n = 1, 2, 3, 4$ to denote the different isomorphism types of $\mathrm{Tree}_{\Gamma_f}(x)$ for periodic x. Additionally, we define $\mathfrak{T}_0$ to denote the isomorphism type of the trivial rooted tree, consisting of a single vertex without arcs.

Now that we have a full understanding of the rooted trees above vertices in Γ_f, let us turn to the determination of the f-periodic points and to the construction of a CRL-list for f. Following the discussion in Section 3.1, the periodic points of f are the field element 0 as well as all periodic points of f in the unique periodic coset C_0, which we already identified above (using Lemma 2.1.14) to be precisely those $x \in \mathbb{Z}/51\mathbb{Z}$ with $x \equiv 1 \pmod 3$, so there are $51/3 = 17$ periodic points in C_0.

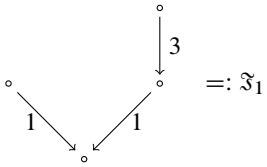
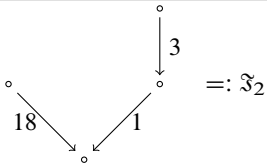
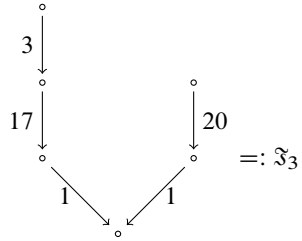
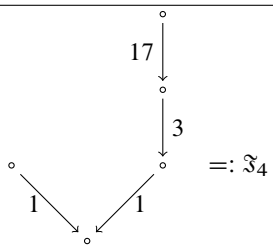
block B of $\mathcal{P}_0$	$\mathrm{Tree}_{\Gamma_f}(x)$ for $x \in B$
$\{x \in \mathbb{Z}/51\mathbb{Z} : x \equiv 1 \pmod 3\} \setminus \{4, 37, 49\}$	 $=: \mathfrak{T}_1$
$\{4\}$	 $=: \mathfrak{T}_2$
$\{37\}$	 $=: \mathfrak{T}_3$
$\{49\}$	 $=: \mathfrak{T}_4$
$B \subseteq \{x \in \mathbb{Z}/51\mathbb{Z} : x \not\equiv 1 \pmod 3\}$	$\mathrm{Tree}_{\Gamma_f}(x, C_3 \cup C_4)$ (see Table 4.2)

Table 4.4. The rooted trees $\mathrm{Tree}_{\Gamma_f}(x)$ for $x \in C_0$.

With such a small number of periodic points, it would be easy to just determine the cycle structure and a CRL-list by brute force, but we would still like to proceed as described in Section 3.1 (and Section 2.3, on which Section 3.1 builds) to illustrate the method.

First, we observe that the nontrivial prime powers of the form $p^{\nu_p(51)}$ are just 3 and 17. By the approach from Section 2.3, we need to determine a CRL-list of each *bijective* reduction of A_0 modulo $p^{\nu_p(51)}$, i.e., here only for the reduction of A_0 modulo 17. We can read off such a CRL-list from Table 2.2. More specifically, since

$$\nu_{17}^{(1)}(1) = 0 \geq 0 = \nu_{17}^{(1)}(9-1),$$

and since ord(9), the multiplicative order of 9 modulo 17, is 8, case 1 in that table with

$$\mathfrak{r} := 3 \quad \text{and} \quad \mathfrak{f} := -\frac{1}{17^0} \cdot \mathrm{inv}_{17}\left(\frac{8}{17^0}\right) = -15 = 2$$

tells us that

$$\{(3^0 17^0 + 2, 8), (3^1 17^0 + 2, 8), (3^0 17^1 + 2, 1)\} = \{(3, 8), (5, 8), (2, 1)\}$$

is a CRL-list of A_0 modulo 17. Modulo 3, the only periodic point of A_0 is 1, so in order to get a CRL-list for A_0 modulo 51, we just map the first entries of the above CRL-list modulo 17 under the function $\Lambda : \mathbb{Z}/17\mathbb{Z} \to \mathbb{Z}/51\mathbb{Z}$ with $\Lambda(x) \equiv x \pmod{17}$ and $\Lambda(x) \equiv 1 \pmod{3}$, which leads to the following CRL-list of A_0 modulo 51:

$$\{(37, 8), (22, 8), (19, 1)\}.$$

We can now describe the isomorphism types of the four connected components of Γ_f as cyclic sequences (necklaces, isomorphism types of necklace graphs) of finite directed rooted trees simply by enumerating the elements on the cycles of f by iteration, then looking up the associated rooted tree isomorphism types in Table 4.4. We note that this is a brute-force approach that is not viable when the number of f-periodic points is large and should then be replaced by the approach described in Section 3.4 instead.

- The connected component of the field element 0 is a single vertex with a loop, corresponding to the following cyclic sequence of rooted tree isomorphism types: $[\mathfrak{I}_0]$.
- Because the cycle of $22 \in \mathbb{Z}/51\mathbb{Z}$ under A_0 is $(22, 46, 7, 13, 16, 43, 31, 25)$, the connected component of the field element $\iota_0(22) = \omega^{5 \cdot 22} = \omega^{110}$ is represented by the cyclic sequence $[\mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1]$.
- Because the cycle of $37 \in \mathbb{Z}/51\mathbb{Z}$ under A_0 is $(37, 28, 49, 34, 1, 10, 40, 4)$, the connected component of the field element ω^{185} is represented by the cyclic sequence

$$[\mathfrak{I}_3, \mathfrak{I}_1, \mathfrak{I}_4, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_2].$$

- Finally, because the cycle of 19 is simply (19), the connected component of the field element ω^{95} is represented by $[\mathfrak{I}_1]$.

As a quick sanity check, we note that $|\mathrm{V}(\mathfrak{I}_1)| = 6$, $|\mathrm{V}(\mathfrak{I}_2)| = 23$, $|\mathrm{V}(\mathfrak{I}_3)| = 91$, and $|\mathrm{V}(\mathfrak{I}_4)| = 57$, so the vertex numbers of the three connected components in $\mathbb{F}_{2^8}^*$ add up to

$$14 \cdot 6 + 23 + 91 + 57 = 255,$$

as they should. One may also verify that these are the same cyclic sequences that were given in our introduction.

4.3 Special case: All A_i are permutations

Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$, given in cyclotomic form (1.1). Let us assume that for each $i = 0, 1, \ldots, d-1$, we have

$$\gcd(r_i, s) = \gcd\left(r_i, \frac{q-1}{d}\right) = 1.$$

An important class of functions to which this applies are the index d cyclotomic mappings of $\mathbb{F}_q$ *of first order* (i.e., those generalized cyclotomic mappings for which all r_i are equal to 1).

By our comments between Remark 1.3 and Definition 1.4, the affine map A_i of $\mathbb{Z}/s\mathbb{Z}$, which encodes the restriction $f_{|C_i}$ in case $a_i \neq 0$, is of the form $z \mapsto r_i z + \text{const}$. Our assumption on the r_i is therefore equivalent to demanding that for each i such that $a_i \neq 0$ (and thus A_i is well defined), the function A_i is an affine *permutation* of $\mathbb{Z}/s\mathbb{Z}$.

Our goal is to describe the functional graph Γ_f, which turns out to be particularly easy. Let us start with the rooted trees.

Lemma 4.3.1. *Let $x \in \mathbb{F}_q = \mathrm{V}(\Gamma_f)$.*

(1) *If $x \neq 0$, and if i denotes the unique index in $\{0, 1, \ldots, d-1\}$ such that $x \in C_i$, then $\mathrm{Tree}_{\Gamma_f}(x)$ is isomorphic to $\mathfrak{T}_i := \mathrm{Tree}_{\Gamma_{\bar{f}}}(i)$.*

(2) *If $x = 0$, then $\mathrm{Tree}_{\Gamma_f}(x) = \sum_{i \in \bar{f}^{-1}(\{d\}) \setminus \{d\}} s\mathfrak{T}_i^+$.*

Proof. Statement (1) can be proved by induction on $h(x) := \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i))$. If $h(x)=0$, then all $\bar{f}$-pre-images of i (if any) are $\bar{f}$-periodic. In particular, x has no f-transient pre-images under f, because each such pre-image would need to lie in a coset C_j, where j is an $\bar{f}$-transient pre-image of i under $\bar{f}$. Indeed, otherwise, i, having an $\bar{f}$-periodic pre-image under $\bar{f}$, is $\bar{f}$-periodic itself. By assumption, we can pick an f-transient pre-image y of x under f in $C_{i'}$, where i' is the unique $\bar{f}$-periodic pre-image of i under $\bar{f}$. If ℓ denotes the cycle length of i under $\bar{f}$, then $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$ represents the restriction of f^ℓ to C_i. Because each A_{i_t} is bijective, so is $\mathcal{A}_i$; in other words, every point in C_i is periodic under $\mathcal{A}_i$ and thus under f (following the discussion in Section 3.1). In particular, x is f-periodic, say with cycle length l. Therefore, $f^{l-1}(x)$ is an f-pre-image of x in $C_{i'}$, as is y. Because $A_{i'}$, which represents the restriction $f_{|C_{i'}} : C_{i'} \to C_i$, is injective, it follows that $f^{l-1}(x) = y$, whence y is f-periodic, contradicting our assumption. The upshot of this discussion is that if $h(x) = 0$, then $\mathrm{Tree}_{\Gamma_f}(x)$ is trivial, as is $\mathfrak{T}_i$.

Now we assume that $h(x) \geq 1$. Let $j_1, j_2, \ldots, j_K$ be the distinct $\bar{f}$-transient pre-images of i under $\bar{f}$. By the argument from the previous paragraph, each f-transient pre-image of x under f must lie in one of the cosets C_{j_t} for $t = 1, 2, \ldots, K$, and

since A_{j_t} is bijective for each t, it follows that x has precisely one (transient) preimage $c_{j_t} \in C_{j_t}$ for each $t = 1, 2, \ldots, K$. Therefore, using the induction hypothesis,

$$\mathrm{Tree}_{\Gamma_f}(x) = \sum_{t=1}^{K} \mathrm{Tree}_{\Gamma_f}(c_{j_t})^+ = \sum_{t=1}^{K} \mathfrak{T}_{j_t}^+ = \sum_{t=1}^{K} \mathrm{Tree}_{\Gamma_{\bar{f}}}(j_t)^+ = \mathrm{Tree}_{\Gamma_{\bar{f}}}(i).$$

For statement (2), let $j_1, j_2, \ldots, j_K$ be the distinct $\bar{f}$-transient children of d in $\Gamma_{\bar{f}}^*$. Equivalently, the j_t are the distinct elements of $\bar{f}^{-1}(\{d\}) \setminus \{d\}$. The f-transient children of $0_{\mathbb{F}_q}$ in Γ_f^* are precisely the points in $\bigcup_{t=1}^{K} C_{j_t}$. Using statement (1), it follows that

$$\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q}) = \sum_{t=1}^{K} \sum_{y \in C_{j_t}} \mathrm{Tree}_{\Gamma_f}(y)^+ \cong \sum_{t=1}^{K} \sum_{y \in C_{j_t}} \mathfrak{T}_{j_t}^+ = \sum_{t=1}^{K} s\mathfrak{T}_{j_t}^+,$$

as required. ∎

Because $\mathrm{Tree}_{\Gamma_f}(x)$ for $x \neq 0$ only depends on the coset C_i in which x lies and can be read off directly from $\Gamma_{\bar{f}}$, we only need to know $\Gamma_{\bar{f}}$ and the cycle structure of f on each coset union $U_i := \bigcup_{t=0}^{\ell-1} C_{i_t}$, where $(i_0, i_1, \ldots, i_{\ell-1})$ with $i = i_0$ is the $\bar{f}$-cycle of i, in order to understand the isomorphism type of Γ_f. This can be achieved using analogous ideas to the ones for the determination of CRL-lists in Section 3.1.

Let us set $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$. Then $\mathcal{A}_i$ is an affine permutation of $\mathbb{Z}/s\mathbb{Z}$, and its cycle type $\mathrm{CT}(\mathcal{A}_i)$ can be read off from [15, Tables 3 and 4]. Moreover,

$$\mathrm{CT}(f_{|U_i}) = \mathrm{BU}_\ell(\mathrm{CT}(\mathcal{A}_i)),$$

where BU_ℓ, the so-called *ℓ-blow-up function*, is the unique $\mathbb{Q}$-algebra endomorphism of $\mathbb{Q}[x_n : n \in \mathbb{N}^+]$ with $\mathrm{BU}_\ell(x_n) = x_{\ell n}$ for all $n \in \mathbb{N}^+$. Say

$$\mathrm{CT}(f_{|U_i}) = x_1^{e_1} x_2^{e_2} \cdots x_{\ell s}^{e_{\ell s}}.$$

Then, viewing isomorphism types of functional graphs as multisets of cyclic sequences (necklaces) of isomorphism types of finite directed rooted trees (with each such sequence encoding the isomorphism type of one connected component), we have the following:

$$\Gamma_f^{(i)} := \Gamma_{f|U_i} = \bigsqcup_{1 \le l \le \ell s, \ell | l} \bigsqcup_{n=1}^{e_l} \{\diamond_{m=1}^{l/\ell}[\mathfrak{T}_{i_0}, \mathfrak{T}_{i_1}, \ldots, \mathfrak{T}_{i_{\ell-1}}]\},$$

and, if $\bar{\mathcal{L}}$ is a CRL-list for $\bar{f}$, then

$$\Gamma_f = \bigsqcup_{(i,\ell) \in \bar{\mathcal{L}}} \Gamma_f^{(i)} \sqcup \left\{\left[\sum_{j \in \bar{f}^{-1}(\{d\}) \setminus \{d\}} s\mathfrak{T}_j^+\right]\right\}.$$

Chapter 5

Algorithmic complexity analysis

The aim of this chapter is to describe algorithms for understanding important aspects of the structure of functional graphs of generalized cyclotomic mappings of finite fields in detail and analyze their complexities. In Section 5.1, we set the ground by describing our computational model, the so-called dual model, in detail and introducing some important auxiliary concepts and results. We note that this dual model consists of carefully keeping track of three distinct parameters – the bit operations, elementary quantum gates and conversions from bits to qubits and vice versa – separately, which is, to the authors' knowledge, a novel approach and may be of independent, wider interest for readers working in quantum complexity analysis. Section 5.2 consists of the proof of Theorem 5.1.9, which provides complexity bounds for three fundamental algorithmic problems and may be considered the main result of this chapter. As mentioned in the introduction, it is an open problem how to encode the overall structure of the functional graph of a generalized cyclotomic mapping compactly; in particular, these results do *not* provide an efficient general algorithm for deciding whether the functional graphs of two given generalized cyclotomic mappings are isomorphic. However, in Section 5.3, we discuss four special cases in which this isomorphism problem can be solved efficiently.

5.1 Framework and auxiliary results

Throughout this chapter, we assume that f is an index d generalized cyclotomic mapping of $\mathbb{F}_q$, given in cyclotomic form (1.1), where either

- each a_i is specified as the field element 0 or as a power of a common, unknown primitive element ω of $\mathbb{F}_q$, or
- we explicitly know the minimal polynomial $P(T)$ over the prime subfield $\mathbb{F}_p$ of such an ω, and the a_i are represented as elements of $\mathbb{F}_p[T]/(P(T))$.

The main goal in this chapter is to analyze the complexities of the following algorithmic problems.

Problem 1. Given f, compute a compact parametrization of a CRL-list $\mathcal{L}$ of f (we note that $|\mathcal{L}|$ equals the number of cycles of f on its periodic points, which may be superpolynomial in $\log q$, so we want to avoid listing $\mathcal{L}$ element-wise).

Problem 2. Given f, compute a partition-tree register of f in the sense of Definition 5.1.2 below.

Problem 3. Given f, a partition-tree register of f, and a pair (r, l) such that $r \in \mathbb{F}_q$ is f-periodic and l is the cycle length of r under f, compute a compact description of the cyclic sequence of rooted tree isomorphism types from formula (3.4) (which characterizes the digraph isomorphism type of the connected component of Γ_f that contains r).

A partition-tree register of f is a standardized way of storing information about the arithmetic partitions $\mathcal{P}_i$ constructed in Section 3.3 and the rooted trees associated with their blocks. To define it, we first introduce the following auxiliary concept.

Definition 5.1.1. A *recursive tree description list* is a finite sequence $(\mathfrak{D}_n)_{n=0,1,\ldots,N}$ of sets that has an associated (unique) ordered sequence $(\mathfrak{T}_n)_{n=0,1,\ldots,N}$ of pairwise distinct, finite rooted tree isomorphism types such that the following hold.

(1) $\mathfrak{T}_0$ is the trivial rooted tree isomorphism type, and $\mathfrak{D}_0 = \emptyset$.

(2) For $n \geq 1$, each rooted tree attached in $\mathfrak{T}_n$ to the root of $\mathfrak{T}_n$ is isomorphic to $\mathfrak{T}_m$ for some $m \in \{0, 1, \ldots, n-1\}$. Moreover, $\mathfrak{D}_n$ is the set of all pairs (m, k_m), where $m \in \{0, 1, \ldots, n-1\}$ is an index for which $\mathfrak{T}_m$ is attached to the root of $\mathfrak{T}_n$ at least once, and k_m is the multiplicity with which it is attached.

In a recursive tree description list, each set $\mathfrak{D}_n$ can be viewed as a compact description of $\mathfrak{T}_n$, referring to the rooted trees attached to the root of $\mathfrak{T}_n$ with their (earlier) indices m, rather than their full descriptions. The idea of encoding isomorphism types of rooted trees via numbers ("tree indices") to get more compact descriptions of larger rooted trees is not new; it appears, for example, in the decision algorithm for isomorphism of directed rooted trees described in [6, Example 3.2 on p. 84]. In contrast to that algorithm, which is linear in the number of vertices, we do not list tree indices m repeatedly, but rather, we specify their multiplicities k_m. In situations such as ours, where entire sets (here: arithmetic partition blocks) of vertices can be dealt with simultaneously, this modification is crucial to ensure the efficiency of our algorithms relative to their smaller input length (which lies in $O(d \log q)$). In implementations, we assume that each $\mathfrak{D}_n$ is represented by an array (ordered list) of pairs (m, k_m), sorted by increasing m. Moreover, m and k_m, both of which are at most q, are to be represented by bit strings of length $\lfloor \log_2 q \rfloor + 1$ (please note, however, that we use other conventions for the related notion of a type-I tree register, introduced in Definition 5.3.2.1 (1)). We observe that with these conventions, all bit strings representing an element (m, k_m) of $\mathfrak{D}_n$ (for some n) have the same length, and the ordering of the elements of $\mathfrak{D}_n$ by increasing m corresponds to the lexicographic ordering of those bit string encodings.

Equipped with the concept of a recursive tree description list, we can define partition-tree registers of generalized cyclotomic mappings of finite fields as follows, using notations introduced in Section 3.3. We note that in this algorithmic chapter, we

frequently identify arithmetic partitions with specific spanning congruence sequences of them.

Definition 5.1.2. Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$. For an $\bar{f}$-periodic index $i \in \{0, 1, \ldots, d-1\}$, we recall that i_t for $t \in \mathbb{Z}$ denotes the unique $\bar{f}$-periodic index in $\{0, 1, \ldots, d-1\}$ such that $(\bar{f}_{|\mathrm{per}(\bar{f})})^t(i) = i_t$. A *partition-tree register of* f is an ordered pair of the form

$$((\mathcal{Z}_i)_{i=0,1,\ldots,d-1}, ((\mathfrak{D}_n, (S_{n,i})_{i=0,1,\ldots,d}))_{n=0,1,\ldots,N})$$

such that the following hold.

(1) For each $i = 0, 1, \ldots, d-1$, $\mathcal{Z}_i$ is the following.
   (a) If i is $\bar{f}$-transient, then $\mathcal{Z}_i = \mathcal{P}_i$, given through a spanning congruence sequence of length $m_i \in \mathbb{N}_0$.
   (b) If i is $\bar{f}$-periodic, then $\mathcal{Z}_i$ is an $(H_i + 2)$-tuple $(\mathcal{X}_{i,h})_{h=-1,0,\ldots,H_i}$ such that
      (i) $\mathcal{X}_{i,-1} = (\theta_{i,h}(x))_{h=1,2,\ldots,H_i}$, and
      (ii) $\mathcal{X}_{i,h}$ for $h = 0, 1, \ldots, H_i$ is (a spanning congruence sequence for) the arithmetic partition $\lambda^h_{i_{-h}}(\mathcal{R}_{i_{-h}})$, of length $n_{i_{-h}} \in \mathbb{N}_0$.

(2) The sequence $(\mathfrak{D}_n)_{n=0,1,\ldots,N}$ is a recursive tree description list, with associated rooted tree isomorphism type sequence $(\mathfrak{T}_n)_{n=0,1,\ldots,N}$, such that the $\mathfrak{T}_n$ are just those rooted tree isomorphism types that are of one of the forms
   (a) $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ for some $\bar{f}$-transient i and some $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$ such that the block $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ is non-empty;
   (b) $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$; or
   (c) $\mathrm{Tree}^{(h)}_i(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$ for some $\bar{f}$-periodic $i \neq d$, some $h \in \{0, 1, \ldots, H_i\}$ and some $\vec{\nu}^{(\mathcal{P}_{i,h})} \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}}$ such that $\mathcal{B}(\mathcal{Q}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h})$, which is the set of all points in $\mathcal{B}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$ of $\mathfrak{h}$-value h, is non-empty.

(3) The objects $S_{n,i}$ satisfy the following.
   (a) If i is $\bar{f}$-transient, then
   $$S_{n,i} = \{\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i} : \mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset \text{ and } \mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \cong \mathfrak{T}_n\}.$$
   (b) If $i = d$, then $S_{n,i} = S_{n,d} \in \{\emptyset, \neg\}$ is the logical sign associated with the truth value of the isomorphism relation $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q}) \cong \mathfrak{T}_n$.
   (c) If $i \neq d$ is $\bar{f}$-periodic, then $S_{n,i} = (S_{n,i,h})_{h=0,1,\ldots,H_i}$, where
   $$S_{n,i,h} = \{\vec{\nu}^{(\mathcal{P}_{i,h})} \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}} : \mathcal{B}(\mathcal{Q}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h}) \neq \emptyset \text{ and } \mathrm{Tree}^{(h)}_i(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})}) \cong \mathfrak{T}_n\}.$$

In implementations, we assume that each set $S_{n,i}$ for $\bar{f}$-transient i, as well as each set $S_{n,i,h}$ for $\bar{f}$-periodic $i < d$ and $h \in \{0, 1, \ldots, H_i\}$, is represented by a lexicographically ordered array of bit strings, where a bit 0 stands for $\emptyset$ and a bit 1 stands for $\neg$. We note that while a partition-tree register for f does not explicitly mention the partitions $\mathcal{P}_i = \mathcal{Q}_{i,H_i}$ for $\bar{f}$-periodic indices $i < d$, it is easy to read off their spanning congruence sequences and associated rooted trees from the register. Namely,

- the concatenation of the congruence sequences in $\mathcal{Z}_i$ spans $\mathcal{P}_i$; and
- by Proposition 3.3.5, the rooted tree associated with a block $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ of $\mathcal{P}_i$ is of the form $\mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$ for suitable $h \in \{0, 1, \ldots, H_i\}$ and

$$\vec{\nu}^{(\mathcal{P}_{i,h})} \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}}.$$

  The relevant parameters h and $\vec{\nu}^{(\mathcal{P}_{i,h})}$ can be read off from the logical sign tuple $\vec{\nu}^{(\mathcal{P}_i)}$ that characterizes the block of $\mathcal{P}_i$.

Before we proceed with the actual complexity analysis of Problems 1–3, we make some comments, starting with a discussion of our computational model.

As was already hinted at in Section 2.4, in order to even stand a chance of achieving polynomial runtime for our algorithms, we need quantum computers at least for certain subtasks, such as whenever a modular multiplicative order or a discrete logarithm needs to be computed. That being said, we only relegate certain well-defined tasks, for which efficient quantum algorithms are already known, to quantum computers, while the rest of our algorithms can be performed on a classical computer. Therefore, we use the following two computational models:

- a bit operation model with several kinds of queries for the tasks for which no efficient algorithms are known on a classical computer. In this model, which we henceforth refer to as the *query model* (and an algorithm in that model is a *query algorithm*), the complexity is measured as a tuple that tracks the amount of bit operations used outside queries and the amount of times each kind of query is called for;
- a model in which classical and quantum computers are used in tandem and can "feed" their outputs to each other; we refer to algorithms built like that as *dual algorithms*, and to the model as the *dual model*.

For the quantum side of the computations in the dual model, we specifically use the quantum *circuit* model, so whenever we speak of *quantum complexity*, we mean quantum *(elementary) gate* complexity. All quantum algorithms which we use are based on Shor's seminal paper [68], and all of them are *Las Vegas algorithms*, i.e., their runtime on a given input varies randomly, and their specified bit operation cost, gate complexity and number of conversions from bits to qubits and vice versa are to be understood as *expected* values. This also means that as a whole, all of our dual

algorithms are Las Vegas algorithms, and all parts of their specified complexities are expected values only.

On the other hand, for the "classical" side of the computations in either model, we use a bit operation model based on random memory access in the vein of [6, Section 1.2], in which memory access takes $O(N)$ bit operations if N is the bit length of the address (index) of the memory register that needs to be accessed. For example, accessing the stored value of a variable y_k, where k is a non-negative integer takes $O(\log k)$ bit operations – the entire memory address consists of a bit encoding for the letter "y" (which is assumed to be of length $O(1)$), concatenated with the standard binary representation of k. We thus assume that "jumping" to a place in memory after its address has been scanned is free. In addition to accessing memory registers by reading in their addresses, we also assume that we can save certain positions within a register through placing pointers (of which we have a finite amount, though we do not specify a concrete bound on their number), which enables us to jump back to that specific position (bit) in memory at a cost of $O(1)$ bit operations. Moreover, we assume that it takes $O(1)$ bit operations to move to a neighboring position in memory, including to the next entry of an array. We refer to the classical part of our complexity as *classical complexity* or (synonymously) *bit operations*.

Now, it is well known (see, e.g., [84, Section IV.3]) that each classical circuit has an equivalent quantum circuit in which the number of elementary gates is only larger by at most a constant factor. Based on this, it may seem tempting to just use circuits for both kinds of computations in the dual model, so that the classical part could be subsumed (without changing the Landau O-class of the gate complexity) in the quantum part, and it appears that this is the usual approach for quantum complexity analysis. For example, in [37, Sections 7.3 and 7.4], the complexity analysis of specific quantum algorithms (i.e., those that do not involve operations in a black-box group) only consists of counting the involved number of quantum gates, while the bit operation (or classical gate) cost of pre- and post-processing is ignored. For the algorithms in [37, Sections 7.3 and 7.4], this is perfectly fine, as that classical cost is a big-O of the quantum gate count regardless of whether bit operations or classical gates are used for measurement. However, our algorithms do involve a significantly larger classical cost than quantum gates, as can already be seen in the complexity bounds from Lemma 5.1.6; we note that while these are essentially algorithms from [37, Sections 7.3 and 7.4], they do end up with a relatively large classical cost if one wants them to be Las Vegas algorithms (due to the use of the AKS primality test). Hence, keeping track of the classical cost (whether bit operations or classical gates) and quantum gates separately seems natural, especially since the actual time cost of each quantum gate in a large-scale physical implementation of a quantum computer is not known at this point.

As for why we use bit operations (and not gates) in the classical part of our computations, we note that our algorithms for solving Problems 2 and 3 involve a copious

amount of "bookkeeping", i.e., memory access, and in any of the two circuit models, memory access is generally costly. Indeed, let us assume that, say, in the classical circuit model, we wish to access the value of a previously computed variable $y_k \in \{0, 1\}$, where the index $k \in \{1, \ldots, N\}$ is also a result of an earlier computation. When building the circuit, we do not know a priori which of the associated N wires carries the relevant information, and so this needs to be processed via a subcircuit that takes as input those N wires and the $O(\log N)$ wires carrying (the bit representation of) k. But each elementary gate only accepts $O(1)$ input bits, so the said subcircuit performing the memory access must consist of at least cN elementary gates for some constant $c > 0$, as opposed to the $O(\log N)$ cost of memory access for the analogous problem in our chosen bit operation model.

When communication between the classical and quantum part of a dual algorithm happens, classical bit strings $\vec{x} \in \{0, 1\}^N$ need to be converted into the corresponding qubit registers $|\vec{x}\rangle$ and vice versa. In our algorithms dealing with generalized cyclotomic mappings of $\mathbb{F}_q$, the bit length N is in $O(\log q)$ for each such conversion. Because it is not clear how costly such conversions are, it is of interest to count them separately (for both conversion directions together) in what we call the *conversion complexity* of the corresponding dual algorithm. The copying of converted information over to the next classical computer or quantum circuit, respectively, as well as the measurement taken at the end of a quantum circuit, are considered a part of the respective conversion process, and we do not track their cost separately. For standardization purposes, we assume that both the original input and final output of a dual algorithm are classical bit strings. In particular, the complexity of a pure quantum algorithm that is viewed as a dual algorithm involves two conversions (one each at the beginning and end of the algorithm) in addition to the quantum gate count.

Let us also talk about Grover's quantum algorithm for unstructured database search from [26]. This algorithm is famous for providing a quadratic speedup over the classical linear search algorithm, and given the aforementioned copious amount of bookkeeping in our algorithms, it seems natural to use it. However, there are some subtleties to take into account here, which ultimately led the authors to decide against the inclusion of Grover's algorithm in our analysis. The usual complexity analysis for Grover's algorithm assumes that the list to be searched (or rather, the associated characteristic function χ for the piece of information we want to find in the list) is given as a certain unitary operator U_χ, called *phase inversion*, which is subsequently used as a part of the quantum circuit for the algorithm and treated as an oracle. The celebrated Grover complexity of $O(\sqrt{N})$ for searching a list of length N refers to the number of times U_χ (and another, so-called phase shift operator) is applied before the final measurement. However, we are interested in gate complexities, so the gate complexity of U_χ needs to be included as an additional factor. Now, χ could be any function $\{0, 1, \ldots, N-1\} \to \{0, 1\}$ (the oft-used assumption that $\chi(j) = 1$ for a unique index j does not apply to our case), which we may also view as a (partial)

Boolean function in $n = \lfloor \log_2 N \rfloor$ variables. This means that in order for the quantum (gate) complexity of Grover's algorithm to "beat" the bit operation complexity of linear search, the worst-case quantum gate complexity of an n-variable Boolean function would need to be in $o(2^{n/2})$, and it is not clear whether this holds. We do note that it is known that the worst-case *classical* gate complexity of a Boolean function in n variables is of order of magnitude $2^n/n$ (Shannon, [67, Theorems 6 and 7 on pp. 76f.]), and that it is only a certain power away from the worst-case quantum gate complexity of an n-variable Boolean function (Beals et al., [12]).

We observe that our treatment of quantum algorithms in the dual model is idealized in the sense that we ignore the possibility of errors due to hardware failure and quantum noise. Like many authors, we do so relying on the celebrated quantum threshold theorem, the morale of which is that once the failure rate per elementary gate can be pushed beneath a certain, constant threshold, arbitrarily robust quantum algorithms can be constructed at little extra cost compared to their idealized counterparts. This theorem dates back to a paper of Shor [69], though the version stated there is weaker than what the theorem is known as today. Several variants of the stronger version (depending on the error model used) were proved independently by Aharonov and Ben-Or [4], Knill, Laflamme and Zurek [39], and Kitaev [38], respectively. The survey [25], in which the theorem is stated as Theorem 10, provides a unified proof of it.

The preceding discussion motivates the following definition of the notions of algorithmic complexity which our results in this chapter refer to.

Definition 5.1.3. We introduce the following concepts and notations.

(1) We denote by $\{0,1\}^{<\infty}$ the set of all finite bit strings. Formally,

$$\{0,1\}^{<\infty} = \bigcup_{n \in \mathbb{N}_0} \{0,1\}^n.$$

(2) An *algorithmic problem* is a function $\mathfrak{L}$ defined on a subset $\mathfrak{L}_{\text{in}}$ of $\{0,1\}^{<\infty}$ and mapping each bit string $\vec{x} \in \mathfrak{L}_{\text{in}}$ to some non-empty finite subset $\mathfrak{L}(\vec{x}) \subseteq \{0,1\}^{<\infty}$.

(3) In the situation of statement (2), the elements of $\mathfrak{L}_{\text{in}}$ are called the *admissible inputs for* $\mathfrak{L}$, and for each $\vec{x} \in \mathfrak{L}_{\text{in}}$, the elements of $\mathfrak{L}(\vec{x})$ are called the *admissible outputs for* $\vec{x}$ *(with respect to* $\mathfrak{L}$*)*.

(4) Let $\mathfrak{L}$ be an algorithmic problem, and let $y, y_1, y_2, \ldots, y_n$ be non-negative real parameters associated with the admissible inputs for $\mathfrak{L}$; formally, y and the y_j are functions $\mathfrak{L}_{\text{in}} \to [0,\infty)$.

(a) A tuple $\vec{\mathcal{C}}^{(\text{qry})} = (\mathcal{C}_{\text{class}}, \mathcal{C}_{\text{fdl}}, \mathcal{C}_{\text{mdl}}, \mathcal{C}_{\text{mord}}, \mathcal{C}_{\text{prt}})$ each entry of which is a function $[0,\infty)^n \to [0,\infty)$ is called a *y-bounded query complexity of* $\mathfrak{L}$

(with respect to $y_1, \ldots, y_n$) if there is a query algorithm which on each input $\vec{x} \in \mathfrak{L}_{\text{in}}$ produces an admissible output for $\vec{x}$ using

- $O(\mathcal{C}_{\text{class}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ bit operations outside the queries listed below;
- $O(\mathcal{C}_{\text{fdl}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ queries to compute a discrete logarithm in a finite field of size at most $y(\vec{x})$;
- $O(\mathcal{C}_{\text{mdl}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ queries to compute, for given $x, z \in (\mathbb{Z}/m\mathbb{Z})^*$, where $m < y(\vec{x})^2$, the modular discrete logarithms $\log_x^{(k)}(z)$, where
$$k \in \{m, p^{v_p(m)} : p \mid m\},$$
outputting a list consisting of the pair $(m, \log_x^{(m)}(z))$ and the quadruples
$$(p, v_p(m), p^{v_p(m)}, \log_x^{(p^{v_p(m)})}(z))$$
for all primes $p \mid m$;
- $O(\mathcal{C}_{\text{mord}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ queries to compute, for a given unit $x \in (\mathbb{Z}/m\mathbb{Z})^*$, where $m < y(\vec{x})^2$, the multiplicative orders $\text{ord}_k(x)$, where $k \in \{m, p^{v_p(m)} : p \mid m\}$, outputting a list consisting of the pair $(m, \text{ord}_m(x))$ and the quadruples
$$(p, v_p(m), p^{v_p(m)}, \text{ord}_{p^{v_p(m)}}(x))$$
for all primes $p \mid m$;
- $O(\mathcal{C}_{\text{prt}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ queries to find a primitive root $\text{r}^{(p)}$ modulo each odd prime power divisor $p^{v_p(m)} > 1$ for some integer $m < y(\vec{x})$, outputting the corresponding list of quadruples
$$(p, v_p(m), p^{v_p(m)}, \text{r}^{(p)}).$$

(b) A *y-bounded Las Vegas dual complexity for* $\mathfrak{L}$ is a triple
$$\vec{\mathcal{C}}^{(\text{LV})} = (\mathcal{C}_{\text{class}}, \mathcal{C}_{\text{quant}}, \mathcal{C}_{\text{conv}})$$
each entry of which is a function
$$[0, \infty)^n \to [0, \infty)$$
such that there is an (idealized) dual algorithm which on each input $\vec{x} \in \mathfrak{L}_{\text{in}}$ terminates after an expected number of

- $O(\mathcal{C}_{\text{class}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ bit operations,
- $O(\mathcal{C}_{\text{quant}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ elementary quantum gates, and

- $O(\mathcal{C}_{\mathrm{conv}}(y_1(\vec{x}), y_2(\vec{x}), \ldots, y_n(\vec{x})))$ conversions of length $O(\log y(\vec{x}))$ bit strings into qubit registers and of length $O(\log y(\vec{x}))$ qubit registers into bit strings,

producing an admissible output for $\vec{x}$.

In our algorithms, the value of $y(\vec{x})$ from Definition 5.1.3 is always equal to the corresponding field size q. While our definition of the query model does not explicitly include integer factorization queries, they are subsumed in either of modular discrete logarithm queries or multiplicative order queries. Indeed, in order to factor $m \in \mathbb{N}^+$ with $m < y(\vec{x})^2$, one can simply make the query to compute the multiplicative order modulo m of $x := 1 \in (\mathbb{Z}/m\mathbb{Z})^*$. Beside the pair $(m, 1)$, the resulting output consists of the quadruples $(p, \nu_p(m), p^{\nu_p(m)}, 1)$, where p ranges over the prime divisors of m, from which it is straightforward to read off the prime factorization of m. Likewise, one could make a modular discrete logarithm query with $x := y := 1$.

The assumption from Definition 5.1.3 (2) that each admissible input for $\mathfrak{L}$ should only have finitely many admissible outputs is without loss of generality for our analysis. It simplifies the formulation of Lemma 5.1.4 below, which is straightforward to prove and basically states that query complexities behave additively with respect to composition of algorithmic problems, which is defined as follows. If $\mathfrak{L}$ and $\mathfrak{L}'$ are algorithmic problems such that $\mathfrak{L}(\vec{x}) \subseteq \mathfrak{L}'_{\mathrm{in}}$ for each $\vec{x} \in \mathfrak{L}_{\mathrm{in}}$, then the *composition of* $\mathfrak{L}$ *and* $\mathfrak{L}'$, written $\mathfrak{L}\mathfrak{L}'$ or $\mathfrak{L}' \circ \mathfrak{L}$, is the algorithmic problem with input set $\mathfrak{L}_{\mathrm{in}}$ that is defined via $(\mathfrak{L}' \circ \mathfrak{L})(\vec{x}) := \bigcup \mathfrak{L}'(\mathfrak{L}(\vec{x}))$ (the union of the sets that make up the element-wise image of $\mathfrak{L}(\vec{x})$ under $\mathfrak{L}'$, i.e., all finite bit strings which are admissible outputs, with respect to $\mathfrak{L}'$, for some admissible output for $\vec{x}$ with respect to $\mathfrak{L}$).

Lemma 5.1.4. *Let $\mathfrak{L}$ and $\mathfrak{L}'$ be algorithmic problems such that $\mathfrak{L}(\vec{x}) \subseteq \mathfrak{L}'_{\mathrm{in}}$ for each $\vec{x} \in \mathfrak{L}_{\mathrm{in}}$, and let $y, y_1, y_2, \ldots, y_n$, respectively, $y', y'_1, y'_2, \ldots, y'_{n'}$ be non-negative real parameters that are associated with the admissible inputs of $\mathfrak{L}$, respectively, $\mathfrak{L}'$. For $j' = 1, 2, \ldots, n'$, we define*

$$\begin{aligned} y_{n+j'} : \mathfrak{L}_{\mathrm{in}} &\to [0, \infty), \\ \vec{x} &\mapsto \max\{y'_{j'}(\vec{y}) : \vec{y} \in \mathfrak{L}(\vec{x})\}. \end{aligned}$$

Because the composition $\mathfrak{L}\mathfrak{L}'$ has input set $\mathfrak{L}_{\mathrm{in}}$, we may view each y_j for $1 \leq j \leq n + n'$ as a parameter for $\mathfrak{L}\mathfrak{L}'$. Moreover, we define

$$\begin{aligned} y^+ : \mathfrak{L}_{\mathrm{in}} &\to [0, \infty), \\ \vec{x} &\mapsto \max\{y(\vec{x}), y'(\vec{y}) : \vec{y} \in \mathfrak{L}(\vec{x})\}. \end{aligned}$$

Finally, we let $\vec{\mathcal{C}}^{(\mathrm{qry})}$, respectively, $\vec{\mathcal{C}'}^{(\mathrm{qry})}$, be a y-bounded query complexity for $\mathfrak{L}$ with respect to $y_1, \ldots, y_n$, respectively, a y'-bounded query complexity for $\mathfrak{L}'$ with respect to $y'_1, \ldots, y'_{n'}$. For $k = 1, \ldots, 5$, we denote by $\mathcal{C}_k$, respectively, $\mathcal{C}'_k$, the k-th

entry of $\vec{\mathcal{C}}^{(\mathrm{qry})}$*, respectively,* $\vec{\mathcal{C}'}^{(\mathrm{qry})}$*. Then, defining*

$$\begin{aligned}&\mathcal{C}_k^+ : [0,\infty)^{n+n'} \to [0,\infty),\\ &(z_1,\dots,z_{n+n'}) \mapsto \mathcal{C}_k(z_1,\dots,z_n) + \mathcal{C}'_k(z_{n+1},\dots,z_{n+n'}),\end{aligned}$$

the tuple $\vec{\mathcal{C}}^{+(\mathrm{qry})} := (\mathcal{C}_1^+,\dots,\mathcal{C}_5^+)$ *is a* y^+*-bounded query complexity for* $\mathfrak{L}\mathfrak{L}'$ *with respect to* $y_1,\dots,y_{n+n'}$*.*

On the other hand, for each given algorithmic problem $\mathfrak{L}$ and non-negative real parameter y associated with the admissible inputs for $\mathfrak{L}$, a y-bounded Las Vegas dual complexity for $\mathfrak{L}$ can be derived from a y-bounded query complexity for $\mathfrak{L}$ as long as y can be bounded in terms of the other parameters y_j; see Lemma 5.1.7 below.

As usual, when specifying complexities in a concrete situation, we identify functions with their defining terms. For example, if d and q are the relevant parameters associated with our inputs, we may specify a q-bounded Las Vegas dual complexity as

$$(d^3 \log q,\ d \log^{2+o(1)} q,\ \log^{1+o(1)} q),$$

rather than introduce names for the functions in the three components. In this context, we also note that $\log^k x$ always denotes the arithmetic power $(\log x)^k$, *not* the function value at x of the k-fold iterate of log. We always spell iterated logarithms out ($\log\log$, $\log\log\log$, etc.).

The following lemma, which is used throughout this chapter, provides the complexities of some fundamental algorithmic problems. For more information, we refer to the classical books [11, 75].

Lemma 5.1.5. *The following hold.*

(1) *Addition and subtraction of integers of absolute value less than* m*, as well as addition and subtraction modulo* m *cost* $O(\log m)$ *bit operations each.*

(2) *Addition and subtraction in the finite field* $\mathbb{F}_q$ *cost* $O(\log q)$ *bit operations each.*

(3) *Multiplication of positive integers less than* m*, multiplication modulo* m *and division of positive integers less than* m *with remainder each respectively cost* $O(\log^{1+o(1)} m)$ *bit operations.*

(4) *Let* $x \in (\mathbb{Z}/m\mathbb{Z})^*$*. The computation of* $\mathrm{inv}_m(x)$*, the multiplicative inverse of* x *modulo* m*, costs* $O(\log^{1+o(1)} m)$ *bit operations.*

(5) *Multiplication, multiplicative inversion and division in the finite field* $\mathbb{F}_q$ *each have a bit operation cost of* $O(\log^{1+o(1)} q)$*.*

(6) *Let* $x \in \mathbb{Z}/m\mathbb{Z}$ *and* $e \in \mathbb{N}_0$*. The computation of the power* x^e *modulo* m *has a bit operation cost of* $O(\log(e)\log^{1+o(1)} m)$*.*

(7) *Let $x \in \mathbb{F}_q$ and $e \in \mathbb{N}_0$. The computation of x^e costs $O(\log(e)\log^{1+o(1)} q)$ bit operations.*

(8) *The computations of the* gcd *and* lcm *of two positive integers that are at most m cost $O(\log^{1+o(1)} m)$ bit operations each.*

(9) *Checking deterministically whether a given positive integer m is a prime has a bit operation cost of $O(\log^{6+o(1)} m)$.*

(10) *An array of n bit strings, each of length k, can be lexicographically sorted with $O(kn \log n)$ bit operations.*

(11) *We assume given two lexicographically sorted arrays of bit strings (not necessarily all of the same length) and consider the algorithmic problem of finding the lexicographically sorted version of their concatenation (i.e., the problem of* merging *those sorted arrays). For $j = 1, 2$, say the j-th array has N_j entries, and the sum of the bit lengths of the strings stored in it is $l^{(j)}_{\text{total}}$. Then those two sorted arrays can be merged within $O(N_1 + N_2 + l^{(1)}_{\text{total}} + l^{(2)}_{\text{total}})$ bit operations (and thus within $O(l^{(1)}_{\text{total}} + l^{(2)}_{\text{total}})$ bit operations if all bit strings in question are non-empty).*

Proof. For statement (1), it is well known (and easy to check) that using the schoolbook algorithms for addition and subtraction yields the specified complexities.

For statement (2), let $q = p^m$. We refer to [51, Table 2.8 on p. 84], and note that the only $(\mathbb{Z}/p\mathbb{Z})$-operations involved in an addition/subtraction in $\mathbb{F}_q$ are modular additions/subtractions. Therefore, statement (1) implies that the cost of addition and subtraction in $\mathbb{F}_q$ is in $O(m \log p) = O(\log q)$, as required.

For statement (3), it follows from the Schönhage–Strassen algorithm [65] or the (slightly faster) algorithm [30] by Harvey and van der Hoeven that the multiplication of two positive integers less than m costs $O(\log^{1+o(1)} m)$ bit operations. Moreover, integer division with remainder also costs $O(\log^{1+o(1)} m)$ bit operations if the Newton–Raphson algorithm is used for it; see [2, Section 1.3]. Multiplication modulo m can be done by performing a (non-modular) multiplication of the two integers in question (resulting in a number with $O(\log(m^2)) = O(\log m)$ bits), followed by a modular reduction (which is a part of division with remainder). In total, multiplication modulo m thus also only requires $O(\log^{1+o(1)} m)$ bit operations.

For statement (4), we note that inversion modulo m can be done with the Extended Euclidean algorithm, which takes $O(\log^{1+o(1)} m)$ bit operations when using an accelerated variant of it due to Schönhage (based on earlier ideas of Knuth) [64]; see also [83], which provides a generalization of this and may be more accessible due to being written in English.

For statement (5), we note that $\mathbb{F}_q$ is given as $(\mathbb{Z}/p\mathbb{Z})[T]/(P(T))$, where $P(T)$ is a monic primitive irreducible polynomial of degree $m := \log_p q$. In order to multiply two elements $Q_1(T) + (P(T))$ and $Q_2(T) + (P(T))$ of $\mathbb{F}_q$, one computes the

polynomial product $Q_1(T)Q_2(T) \in (\mathbb{Z}/p\mathbb{Z})[T]$, then determines its remainder upon division by $P(T)$. As observed in [76, second paragraph in Section 2], fast methods for multiplication of polynomials over $\mathbb{Z}/p\mathbb{Z}$ of degree at most n, as well as for divisions with remainder of such polynomials, take $O(n^{1+o(1)})$ operations (additions, subtractions, multiplications, multiplicative inversions) in $\mathbb{Z}/p\mathbb{Z}$. This corresponds to a bit operation cost of $O(n^{1+o(1)} \log^{1+o(1)} p)$ by statements (1) and (3). It follows that the computation of $Q_1(T)Q_2(T)$, and the subsequent computation of its remainder modulo $P(T)$, both take $O(m^{1+o(1)} \log^{1+o(1)} p) = O(\log^{1+o(1)} q)$ bit operations, as needed for the asserted complexity bound on multiplication in $\mathbb{F}_q$ to hold.

Now, because a division in $\mathbb{F}_q$ consists of a multiplicative inversion followed by a multiplication, it suffices to argue that the bit operation cost of multiplicative inversion in $\mathbb{F}_q$ is in $O(\log^{1+o(1)} q)$ to conclude the proof of this statement. Assuming that $P(T) \nmid Q(T)$, the multiplicative inversion of $Q(T)$ modulo $P(T)$ may be performed by writing $1 = \gcd(P(T), Q(T))$ as a $(\mathbb{Z}/p\mathbb{Z})[T]$-linear combination of $P(T)$ and $Q(T)$, and reducing the scalar of $Q(T)$ in this linear combination modulo $P(T)$. The algorithm described in [6, Section 8.9] uses $O(\log^{1+o(1)}(p^m) \cdot \log m) = O(\log^{1+o(1)} q)$ bit operations to compute $\gcd(P(T), Q(T))$ (according to [6, Theorem 8.19]). In the process, one may store the (2×2)-matrices $R_1, R_2, \ldots, R_K$ (with coefficients in $(\mathbb{Z}/p\mathbb{Z})[T]$) that are output (in the listed order) by the $K \in O(\log m)$ calls of the HGCD procedure from [6, Figure 8.7 on p. 304]. Let $P_0(T) = P(T)$, $P_1(T) = Q(T)$, $P_2(T), \ldots, P_N(T) = \gcd(P(T), Q(T)) = 1$ be the successive remainders appearing in the classical, "slow" version of the Euclidean algorithm applied to $(P(T), Q(T))$, and for $t \in \{0, 1, \ldots, N-1\}$, let $Q_t(T)$ be the quotient of the polynomial division of $P_t(T)$ by $P_{t+1}(T)$, which satisfies $\deg Q_t(T) = \deg P_t(T) - \deg P_{t+1}(T)$. By [6, statement on p. 303 that the output of HGCD is of the form R_{0j}, and definition of R_{ij} before Example 8.10 on p. 302], each of the matrices R_j is a product of matrices of the form

$$\begin{pmatrix} 0 & 1 \\ 1 & -Q_t(T) \end{pmatrix}$$

for pairwise distinct t. Therefore, each entry of $R_j(T)$ is a polynomial in $(\mathbb{Z}/p\mathbb{Z})[T]$ of degree at most

$$\sum_{t=0}^{N-1} \deg Q_t(T) = \sum_{t=0}^{N-1} (\deg P_t(T) - \deg P_{t+1}(T)) = \deg P(T) - \deg 1 = \deg P(T).$$

Moreover, by [6, Lemma 8.5 (a)], for each $k \in \{1, 2, \ldots, K\}$, one has

$$R_k R_{k-1} \cdots R_1 \cdot \begin{pmatrix} P(T) \\ Q(T) \end{pmatrix} = \begin{pmatrix} P_j(T) \\ P_{j+1}(T) \end{pmatrix}$$

for some $j = j(k) \in \{0, 1, \ldots, N-1\}$, and specifically

$$R_K R_{K-1} \cdots R_1 \cdot \begin{pmatrix} P(T) \\ Q(T) \end{pmatrix} = \begin{pmatrix} P_{N-1}(T) \\ P_N(T) \end{pmatrix} = \begin{pmatrix} P_{N-1}(T) \\ 1 \end{pmatrix}.$$

This latter equality yields an expression of 1 as a linear combination of $P(T)$ and $Q(T)$, in which the (reduction modulo $P(T)$ of the) scalar of $Q(T)$ is equal to the (reduction modulo $P(T)$ of the) lower right coefficient of the (2×2)-matrix $R_K R_{K-1} \cdots R_1$. It takes $O(K) \subseteq O(\log m)$ additions, multiplications and divisions with remainder in $(\mathbb{Z}/p\mathbb{Z})[T]$ to compute this matrix product, which also corresponds to a bit operation cost of $O(\log m \cdot \log^{1+o(1)} p^m) = O(\log^{1+o(1)} q)$, as required.

For statement (6), we note that using "Square and Multiply", the power $x^e \bmod m$ can be computed with $O(\log e)$ multiplications modulo m, so statement (3) yields the claim.

For statement (7), the proof is analogous to the one for statement (6), but using statement (5) in place of statement (3).

For statement (8), one may use the Euclidean algorithm to compute a greatest common divisor and refer to [64] or [83]. Moreover, $\operatorname{lcm}(x, y) = xy/\gcd(x, y)$, so statement (3) completes the proof of this claim.

For statement (9), the asserted complexity is achieved by a variant of the AKS primality test devised by Lenstra and Pomerance, see [3, 45].

For statement (10), we refer the reader to [6, Algorithm 3.1 on pp. 78f.], observing that the variable m from that algorithm has the value 2 in our situation. We note that while [6, Theorem 3.1 on p. 79] states that this algorithm costs $O(kn)$ bit operations, this uses an assumption which our computational model does not share. Namely, [6, Algorithm 3.1 on pp. 78f.] uses a "pointer" for each bit string, which must not be confused with the way we use that word. In our model, a pointer is a short-cut to jump to a previously saved point in memory using only $O(1)$ bit operations, and we may only use $O(1)$ of these pointers (i.e., the number of pointers used must not tend to ∞ as the input length tends to ∞). On the other hand, in [6, Algorithm 3.1 on pp. 78f.], the word "pointer" appears to denote what we would call the memory address of the respective bit string. It is stated explicitly in [6, Algorithm 3.1 on pp. 78f.] that the authors of that book assume that a pointer in their sense can be processed (i.e., stored and used to jump to the respective bit string) within $O(1)$ bit operations. However, in our model, since n of these memory addresses are needed, it takes $O(\log n)$ bit operations to process an address, which leads to the additional factor $\log n$ in our cost.

For statement (11), we observe that it is easy to prove that the merging algorithm [40, Algorithm M on p. 158] achieves this complexity, as long as pointers (in our sense of the word) are used to immediately jump back to saved positions in the arrays, which avoids additional logarithmic factors in the complexity. In fact, in the discussion from [40, p. 159], it is stated that the achieved complexity is in $O(N_1 + N_2)$, but

this uses the assumption that each stored bit string has constantly bounded bit length (causing $l^{(j)}_{\text{total}} \in O(N_j)$ for $j = 1, 2$). ∎

The next lemma essentially provides the Las Vegas dual complexities of the query problems from our query model. It is used to translate query complexities into Las Vegas dual complexities; see Lemma 5.1.7 below.

Lemma 5.1.6. *The following hold.*

(1) *The prime factorization of the positive integer m can be performed with a Las Vegas dual algorithm with m-bounded complexity*

$$(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m).$$

(2) *The computation of the multiplicative order of $x \in (\mathbb{Z}/m\mathbb{Z})^*$ can be performed with a Las Vegas dual algorithm with (expected) m-bounded dual complexity*

$$(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m).$$

(3) *Let m be a positive integer. For $x, y \in (\mathbb{Z}/m\mathbb{Z})^*$, the modular discrete logarithm $\log_x^{(m)}(y)$ can be computed with a Las Vegas dual algorithm with m-bounded complexity*

$$(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m).$$

(4) *Let q be a prime power. For $x, y \in \mathbb{F}_q^*$, the discrete logarithm $\log_x(y)$ can be computed with a Las Vegas dual algorithm with q-bounded complexity*

$$(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q).$$

(5) *Let p be an odd prime, and k a positive integer. On input (p, k), a primitive root modulo p^k can be found with a Las Vegas dual algorithm with p-bounded complexity* $(\log^{7+o(1)} p, \log^{3+o(1)} p, \log p)$.

Proof. For the proofs of statements (1) and (2), we follow the approach described in [37, Section 7.3], which is originally due to Miller [53] and Shor [68]. A different approach, as kindly pointed out by one of the reviewers, would be to reduce those problems to special instances of the hidden subgroup problem, which would in fact work for all five problems and offer a different explanation for the quantum parts of the complexities (see [37, Corollary 7.5.3]).

For the approach of Miller and Shor, we need to first analyze the complexity of the order-finding algorithm from [37, p. 137], which uses quantum circuits combined with some classical post-processing. It should be noted that this algorithm admits absolute bounds on (i.e., not just expected values of) the different parts of its complexity, but the output is only correct with probability at least 0.399, making this a *Monte Carlo algorithm*, not the Las Vegas algorithm we wish to construct in the end.

First, we need to analyze the complexity of the continued fractions algorithm (it is mentioned in [37, Theorem 7.1.7] that this algorithm has polynomial complexity without specifying the degree). Let us assume given a floating point number (in binary format) that represents the rational number $k/2^n$, where $k \in \{0, 1, \ldots, 2^n - 1\}$. Then there is a sequence of fractions

$$\mathrm{conv}_j(k, 2^n) = \frac{\mathrm{num}_j(k, 2^n)}{\mathrm{den}_j(k, 2^n)}$$

for $j = 1, 2, \ldots, N \in O(n)$, the *(principal) convergents of* $k/2^n$, which are optimal approximations of $k/2^n$ relative to the size of their denominators $\mathrm{den}_j(k, 2^n) \leq 2^n$; for details, we refer the reader to [37, Theorem 7.1.7 and Exercise 7.1.7] and [42, Chapter I]. An important property which we need later is that any reduced integer fraction y/z such that

$$\left| \frac{k}{2^n} - \frac{y}{z} \right| \leq \frac{1}{2z^2}$$

is one of the convergents of $k/2^n$; see [42, Corollary 2 on p. 11]. By [42, Theorem 1 on p. 2], each of the two sequences $(\mathrm{num}_j(k, 2^n))_{j=1,\ldots,N}$ and $(\mathrm{den}_j(k, 2^n))_{j=1,\ldots,N}$ is defined through a simple recursion (involving $O(1)$ integer additions and multiplications in each recursion step) in terms of the so-called *continued fractions coefficients of* $k/2^n$, which are just the integer quotient values in the divisions that occur upon applying the Euclidean algorithm to $(2^n, k)$. In view of [20, Problem 31-2 posed on p. 937] (see also [17] for a worked out solution of this problem using a telescopic sum argument), one can compute and store the continued fraction coefficients of $k/2^n$ using $O(n^2)$ bit operations. Following that, the computation of $\mathrm{num}_j(k, 2^n)$ and $\mathrm{den}_j(k, 2^n)$ for all relevant j takes another $O(n \cdot n^{1+o(1)}) \subseteq O(n^{2+o(1)})$ bit operations by statements (1) and (3) of Lemma 5.1.5.

Having analyzed the continued fractions algorithm, let us now turn to the order-finding algorithm described in [37, p. 137]. In accordance with our notation, we assume that this algorithm is used to find the multiplicative order of x modulo m (in [37], the variable a, respectively, N, is used in place of x, respectively, m). The algorithm starts by computing $m' := \lceil 2 \log_2 m \rceil + 1$, which takes $O(\log m)$ bit operations. We observe that $\mathrm{ord}(x)$, the multiplicative order of x modulo m, is at most $2^{(m'-1)/2}$. After this, we need to initialize two m'-qubit registers, which in our dual model formally takes $O(\log m)$ bit operations for printing the length m' bit strings $0 \cdots 0$ and $0 \cdots 01$, followed by $2 \in O(1)$ conversions of these strings into the corresponding qubit registers $|0\rangle^{\otimes m'}$ and $|0 \cdots 01\rangle$. Steps 4–6 of the algorithm in [37, p. 137] are applications of quantum circuits to those registers, which according to the analysis in [37, pp. 138f.] consist of $O(\log^{2+o(1)} m)$ elementary gates. Next, the measurement described in step 7 of [37, p. 137] corresponds to one more conversion in our model (this time from qubits to classical bits), and with high probability, it leads to a "good estimate" (see below) $k_1/2^{m'}$ of a random integer multiple $t/\mathrm{ord}_m(x)$

of $1/\operatorname{ord}_m(x)$, with $t \in \{0, 1, \ldots, \operatorname{ord}_m(x) - 1\}$. Considering step 8 of [37, p. 137] next, we believe that there is a mistake in the formulation of this step, the first sentence of which should in our opinion read (using the notation from there) "Use the continued fractions algorithm to obtain integers $c_1 \geq 0$ and r_1 with $1 \leq r_1 \leq 2^{(n-1)/2}$ such that $|x_1/2^n - c_1/r_1| \leq 1/2^{n+1}$." In any case, this is a formulation that works. Indeed, switching back to our notation, as long as the output $k_1/2^{m'}$ of step 7 is a good estimate of $t/\operatorname{ord}(x)$ for some $t \in \{0, 1, \ldots, \operatorname{ord}(x) - 1\}$, it follows (by the definition of "good estimate" from [37, beginning of Section 7.1.1], see in particular [37, Theorem 7.1.4]) that

$$\left| \frac{t}{\operatorname{ord}(x)} - \frac{k_1}{2^{m'}} \right| \leq \frac{1}{2^{m'+1}} < \frac{1}{2^{m'}} \leq \frac{1}{2\operatorname{ord}(x)^2}$$

whence, as noted above, we have $t/\operatorname{ord}(x) = \operatorname{conv}_j(k_1, 2^{m'})$ for some j by [42, Corollary 2 on p. 11]. By our above analysis of the continued fractions algorithm, one can thus find, using $O(\log^{2+o(1)} m)$ bit operations, an index j and associated values $\operatorname{num}_j(k_1, 2^{m'})$ and $\operatorname{den}_j(k_1, 2^{m'})$ with $\operatorname{den}_j(k_1, 2^{m'}) \leq 2^{(m'-1)/2}$ such that

$$\left| \operatorname{conv}_j(k_1, 2^{m'}) - \frac{k_1}{2^{m'}} \right| = \left| \frac{\operatorname{num}_j(k_1, 2^{m'})}{\operatorname{den}_j(k_1, 2^{m'})} - \frac{k_1}{2^{m'}} \right| \leq \frac{1}{2^{m'+1}},$$

unless we had bad luck with regard to the output of step 7. If so, it makes sense to abandon the computations and output "FAIL", as specified in [37, step 8 on p. 137]. Now, it follows that

$$\begin{aligned} \left| \frac{t}{\operatorname{ord}(x)} - \operatorname{conv}_j(k_1, 2^{m'}) \right| &\leq \left| \frac{t}{\operatorname{ord}(x)} - \frac{k_1}{2^{m'}} \right| + \left| \frac{k_1}{2^{m'}} - \operatorname{conv}_j(k_1, 2^{m'}) \right| \\ &\leq \frac{1}{2^{m'+1}} + \frac{1}{2^{m'+1}} = \frac{1}{2^{m'}} \\ &\leq \min\left(\frac{1}{2\operatorname{ord}(x)^2}, \frac{1}{2\operatorname{den}_j(k_1, 2^{m'})^2} \right). \end{aligned}$$

Using [37, Exercise 7.1.7 (b)], this implies that $t/\operatorname{ord}(x) = \operatorname{conv}_j(k_1, 2^{m'})$, as required for the correctness of the algorithm from [37, p. 137]. Step 9 of [37, p. 137] is just a repetition of steps 1–8, hence does not make a difference for the O-class of the complexity. Finally, steps 10 and 11 of [37, p. 137] take $O(\log^{2+o(1)} m)$ bit operations by statements (6) and (8) of Lemma 5.1.5. In summary, the order-finding algorithm from [37, p. 137] may be viewed as a dual algorithm which, on input $x \in (\mathbb{Z}/m\mathbb{Z})^*$ and m, outputs the multiplicative order of x modulo m with probability at least 39.9% (see [37, Theorem 7.3.2]), and does so taking $O(\log^{2+o(1)} m)$ bit operations and elementary quantum gates each, as well as $O(1)$ conversions of $O(\log m)$-bit strings to $O(\log m)$-qubit registers or vice versa. As noted in [37, Theorem 7.3.2] (and is clear from step 11), unless the output of that algorithm is "FAIL", it always outputs at

least an integer multiple of $\mathrm{ord}(x)$. This concludes the preparation for the proofs of statements (1) and (2), which we tackle next.

For statement (1), we assume given a positive integer m. We wish to obtain the prime factorization of m. Formally, we wish to output the list of pairs $(p, \nu_p(m))$, where p ranges over the prime divisors of m. First, we describe and analyze a deterministic (classical) algorithm that decides whether m is a power of a single prime p and, if so, outputs $(p, \nu_p(m))$; see also [37, Exercise 7.3.3]. This algorithm is, in turn, based on a deterministic routine that decides whether a given $m \in \mathbb{N}^+$ is a power n^k of a positive integer $n < m$ and, if so, outputs (n, k) for the smallest possible value of $k \geq 2$. We note that if $m = n^k$ for some $n \in \{1, 2, \ldots, m-1\}$ and some $k \in \mathbb{N}^+$, then $k \leq \lfloor \log_2 m \rfloor$. Therefore, we loop over $k = 2, 3, \ldots, \lfloor \log_2 m \rfloor$, and for each fixed value of k, we perform a binary search for n, using the strict monotonicity of the function $x \mapsto x^k$. More specifically, we initialize $n := 2$, and as long as $n^k < m$, we double n until $n^k = m$ or $n^k > m$. In the latter case, we start a binary search between $\frac{n}{2}$ and n. With this approach, the values of the powers n^k which we compute never exceed $2^{\lfloor \log_2 m \rfloor} m \leq m^2$, whence each individual power computation in the process takes

$$O(\log(k) \log^{1+o(1)}(m^2)) = O(\log \log m \log^{1+o(1)} m) = O(\log^{1+o(1)} m)$$

bit operations by statement (6) of Lemma 5.1.5. Because we loop over $O(\log m)$ values of k, and for each k, the binary search for n has $O(\log m)$ iterations, it follows that it takes

$$O(\log m \cdot \log m \cdot \log^{1+o(1)} m) = O(\log^{3+o(1)} m)$$

bit operations to find the minimal working value of k and associated $n = \sqrt[k]{m}$, or see that they do not exist.

Let us now describe a procedure to check whether a given $m \in \mathbb{N}^+$ is a prime power and, if so, write it as such. First, by repeating the above procedure with the loop going backward (i.e., for $k = \lfloor \log_2 m \rfloor, \lfloor \log_2 m \rfloor - 1, \ldots, 2$), one can write $m = n^k$ for the *maximal* $k \in \{1, 2, \ldots, \lfloor \log_2 m \rfloor\}$ such that m has an integer k-th root, also taking $O(\log^{3+o(1)} m)$ bit operations as above. The problem is then reduced to checking whether n is a prime, which takes $O(\log^{6+o(1)} n) \subseteq O(\log^{6+o(1)} m)$ bit operations by statement (9) of Lemma 5.1.5. In summary, we have a deterministic routine with complexity in $O(\log^{6+o(1)} m)$ for deciding whether m is a prime power and, if so, writing it as such.

Shor's general Las Vegas dual approach for factoring $m \in \mathbb{N}^+$ using reduction ideas of Miller is outlined in [37, pp. 132f.]. We start by splitting off the factor $2^{\nu_2(m)}$ from m. Because m is given in its binary representation, this only takes $O(\log^{1+o(1)} m)$ bit operations, accounting for $O(\nu_2(m)) \subseteq O(\log m)$ increases of a counter that remains in $O(\log m)$ throughout (and thus has $O(\log \log m) \subseteq O(\log^{o(1)} m)$ bits). Now, we set $m' := m/2^{\nu_2(m)}$. In the rest of our proof of statement (1), we will

only be dealing with *odd* positive integers. We describe a Las Vegas routine that decides whether a given odd positive integer n is a prime power, then does the following:

- if n is a prime power, it writes n as $p^{v_p(n)}$;
- if n is not a prime power, it finds a factorization of n of the form $n = n' \cdot n''$, where $1 < n', n'' < n$.

We already described above how to decide whether $n = p^{v_p(n)}$ and, if so, write it as such using $O(\log^{6+o(1)} n)$ bit operations, so we start by applying that routine and may henceforth assume that it returned that n is *not* a prime power. Following [37, p. 133], we wish to draw an integer $x \in \{2, 3, \ldots, n-1\}$ uniformly at random. Letting $N := \lfloor \log_2(n-3) \rfloor + 1$, we aim to draw the N-bit integer $y \in \{0, 1, \ldots, n-3\}$ uniformly at random, then set $x := y + 2$. To draw y, we initialize an N-qubit register to $|0\rangle^{\otimes N}$, then pass it through an N-dimensional Hadamard circuit (with elementary gate complexity $N \in O(\log n)$) to get a uniform superposition of all N-bit strings, so that a simple measurement returns (the binary representation of) a random integer in $\{0, 1, \ldots, 2^N - 1\}$. The probability that this integer lies in the range for y is at least $1/2$, so we only need to iterate this procedure an expected number of $O(1)$ times until we get a suitable value for y, using $O(\log n)$ bit operations and elementary quantum gates as well as $O(1)$ conversions to and from $O(\log n)$-bit strings. Following that, we compute $x = y + 2$ and $\gcd(x, n)$, taking $O(\log^{1+o(1)} n)$ bit operations by statements (1) and (8) of Lemma 5.1.5. If $\gcd(x, n) > 1$, we may output the factorization $n = n' \cdot n''$ with $n' = \gcd(x, n)$, taking just another $O(\log^{1+o(1)} n)$ bit operations to compute n'' by division, and are done. Otherwise, x is a (uniformly random) unit modulo n, and we proceed to apply the order-finding routine from [37, p. 137] to get an output o which is either a number or the string "FAIL", and is equal to $\mathrm{ord}(x)$ with probability at least 0.399. If o is "FAIL", we repeat this routine on the same value of x until we get an output that is actually a number (only $O(1)$ repetitions needed by expectancy). Then, if o is *not* even, we abandon this value of x and choose y anew (because we want $2 \mid \mathrm{ord}(x)$, and even if o may not be equal to $\mathrm{ord}(x)$, it is an integer multiple of $\mathrm{ord}(x)$, as was noted above) until we get an x such that either $\gcd(x, n) > 1$ or the associated alleged multiplicative order $o \in \mathbb{N}^+$ is even. This, too, only requires an expected number of $O(1)$ attempts, because for a randomly selected $x \in (\mathbb{Z}/n\mathbb{Z})^*$, the order of x is even with probability at least $1/2$. We then compute $z := x^{o/2} \bmod n$, taking $O(\log^{2+o(1)} n)$ bit operations by statement (6) of Lemma 5.1.5. Because n is not a prime power, we have $\gcd(z-1, n) > 1$ with probability at least $1/2$, so after expectedly $O(1)$ more tries, we will indeed have found a nontrivial factorization of n. Taking into account the complexity of the order-finding routine from [37, p. 137] which we analyzed above, this process expectedly takes $O(\log^{6+o(1)} n)$ bit operations, $O(\log^{2+o(1)} n)$ elementary quantum gates, and $O(1)$ conversions to and from $O(\log n)$-bit strings.

Let us now return to our problem of factoring the odd positive integer $m' = m/2^{v_2(m)}$. Through iteratively applying the factor-finding routine we just described, which needs to be applied $O(\log m)$ times, statement (1) follows (the bit operation cost of some necessary deterministic post-processing, such as adding the exponents of primes appearing in multiple obtained factors, is clearly subsumed under $O(\log^{7+o(1)} m)$).

For statement (2), we assume given a modulus m and a unit $x \in (\mathbb{Z}/m\mathbb{Z})^*$, and we wish to give a *Las Vegas* algorithm that computes $\mathrm{ord}(x)$. For this, we first factor m, using the m-bounded Las Vegas dual complexity $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$ by statement (1). Knowing the factorization of m allows us to compute the Euler totient function value $\phi(m)$ and a factorization thereof within the same m-bounded Las Vegas dual complexity. Now, for each prime $p \mid \phi(m)$, we can work out $v_p(\mathrm{ord}(x))$ as the smallest $v_p \in \{0, 1, \ldots, v_p(\phi(m))\}$ such that

$$x^{p^{v_p}\phi(m)_{p'}} \equiv 1 \pmod{m},$$

where $\phi(m)_{p'} := \phi(m)/p^{v_p(\phi(m))}$. More specifically, we can use a binary search for finding $v_p(\mathrm{ord}(x))$, which according to statement (6) of Lemma 5.1.5 results in a cost of

$$O(\log\log m \cdot \log^{2+o(1)} m) = O(\log^{2+o(1)} m)$$

bit operations for finding $v_p(\mathrm{ord}(x))$ for a single p, hence of $O(\log^{3+o(1)} m)$ bit operations for finding all of these valuations. Finally, we compute $\mathrm{ord}(x)$ itself as the product of all prime powers $p^{v_p(\mathrm{ord}(x))}$, where p ranges over the prime divisors of $\phi(m) \in \mathrm{ord}(x)\mathbb{Z}$. This takes another $O(\log^{3+o(1)} m)$ bit operations, thus proving statement (2).

For the proofs of statements (3) and (4), we need some preparations again. In [37, Section 7.4], a general approach for computing discrete logarithms, working for elements chosen from any black-box group G with a unique encoding of each element, is discussed. For given $x, y \in G$ such that $y = x^t$ for some $t \in \mathbb{Z}$ and the order of x in G is known, this approach returns with high probability the unique $t \in \{0, 1, \ldots, \mathrm{ord}(x) - 1\}$ such that $y = x^t$, which is called the *discrete logarithm (in G) of y with base x*, written $\log_x(y)$. However, if y is *not* a power of x in G, it seems that this approach does not provide a means of confirming this *with certainty*, as is required for the Las Vegas algorithms we desire. This means that in addition to the discrete logarithm algorithm from [37, Section 7.4], we need a Las Vegas routine for checking whether y is a power of x in the first place. We analyze these algorithms one after the other, starting with the routine for computing $\log_x(y)$ in case y is a power of x for $G = (\mathbb{Z}/m\mathbb{Z})^*$ or $G = \mathbb{F}_q^*$. As in Section 2.4, we extend the notation $\log_x(y)$ to arbitrary $x, y \in G$ by setting $\log_x(y) := \infty$ if y is *not* a power of x.

An important observation is that the discrete logarithm algorithm described in [37, p. 144] only works if $\mathrm{ord}(x)$ is a prime (see [37, second paragraph after formula (7.4.2.) on p. 143]). For the general case, we follow "Method 1" from [37, pp. 244f., starting after Corollary A.2.2]. We start by setting $m_0 := 1$ and $\mathrm{rem}_0 := 0$. Then, certainly, $\log_x(y) \equiv \mathrm{rem}_0 \pmod{m_0}$. The aim is to recursively define integers

$$m_1 \mid m_2 \mid \cdots \mid m_N = \mathrm{ord}(x)$$

and $\mathrm{rem}_1, \mathrm{rem}_2, \ldots, \mathrm{rem}_N$ with $\mathrm{rem}_N = \log_x(y)$ such that $\log_x(y) \equiv \mathrm{rem}_k \pmod{m_k}$ throughout. As noted in [37, p. 244, right after Corollary A.2.2], running the algorithm from [37, p. 144] allows us to work out m_{k+1} and rem_{k+1} from $m_k < \mathrm{ord}(x)$ and rem_k with high probability (and no risk of getting incorrect values for them, only "FAIL"). In each step, this involves

- $O(1)$ arithmetic operations covered in statements (1–8) of Lemma 5.1.5, which account for $O(\log^{2+o(1)} m)$ bit operations if $G = (\mathbb{Z}/m\mathbb{Z})^*$, respectively, for $O(\log^{2+o(1)} q)$ bit operations if $G = \mathbb{F}_q^*$;
- $O(\log^{2+o(1)} m)$, respectively, $O(\log^{2+o(1)} q)$, elementary quantum gates; and
- $O(1)$ conversions to and from $O(\log m)$-bit, respectively, $O(\log q)$-bit, strings.

As noted in [37, p. 245], the number N of iterations of this loop is in $O(\log \mathrm{ord}(x))$, and thus in $O(\log m)$, respectively, $O(\log q)$. Therefore, we can compute $\log_x(y)$ in case it is not ∞ and $\mathrm{ord}(x)$ is known using the following m-bounded, respectively, q-bounded, Las Vegas dual complexity:

- $(\log^{3+o(1)} m, \log^{3+o(1)} m, \log m)$ if $G = (\mathbb{Z}/m\mathbb{Z})^*$;
- $(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q)$ if $G = \mathbb{F}_q^*$.

This concludes our preparation for the proofs of statements (3) and (4).

For statement (3), we assume that $x, y \in (\mathbb{Z}/m\mathbb{Z})^*$ are given. In order to compute $\log_x(y)$, we first check whether y is a power of x in the first place. We start by factoring m, which takes m-bounded Las Vegas dual complexity

$$(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$$

by statement (1). For a prime divisor p of m, we set $v_p := v_p(m)$ and $m_p := p^{v_p}$. We wish to compute $\log_x^{(m_p)}(y)$, the discrete logarithm modulo m_p of y with base x, for each prime $p \mid m$, and in the following two paragraphs, we describe how to do so. We use the notation $\mathrm{ord}_n(z)$ to denote the multiplicative order of z modulo n.

First, we assume that $p > 2$. We compute $\mathrm{ord}_{m_p}(x)$ and $\mathrm{ord}_{m_p}(y)$, taking m_p-bounded Las Vegas dual complexity $(\log^{7+o(1)} m_p, \log^{3+o(1)} m_p, \log m_p)$ by statement (2). Because the unit group $(\mathbb{Z}/m_p\mathbb{Z})^*$ is cyclic, we have that y is a power of x modulo m_p if and only if $\mathrm{ord}_{m_p}(y)$ divides $\mathrm{ord}_{m_p}(x)$. By statement (3) of Lemma 5.1.5, it only takes $O(\log^{1+o(1)} m_p)$ bit operations to check this. If this

divisibility does *not* hold, then $\log_x^{(m_p)}(y)$ is ∞, and so is $\log_x^{(m)}(y)$, so we are done. Otherwise, we compute $\log_x^{(m_p)}(y)$ using the Las Vegas routine from [37, p. 144 and Appendix A.2], which takes m_p-bounded Las Vegas dual complexity $(\log^{3+o(1)} m_p, \log^{3+o(1)} m_p, \log m_p)$; we note that at this point, we do know $\operatorname{ord}_{m_p}(x)$ because it was computed beforehand.

Now we assume that $p = 2$. We proceed in a similar manner to when $p > 2$, namely by first checking whether $\log_x^{(m_2)}(y)$ is ∞ and, if not, computing its precise integer value at the cost of an m_2-bounded Las Vegas dual complexity of

$$(\log^{3+o(1)} m_2, \log^{3+o(1)} m_2, \log m_2),$$

or $(\log^{7+o(1)} m_2, \log^{3+o(1)} m_2, \log m_2)$ if $\operatorname{ord}_{m_2}(x)$ has not been computed at that point. Checking whether $\log_x^{(m_2)}(y) = \infty$ is a bit more complicated than for $p > 2$, though, because $(\mathbb{Z}/m_2\mathbb{Z})^*$ is not necessarily cyclic. We may assume that $m_2 > 2$ (otherwise, $\log_x^{(m_2)}(y)$ is simply equal to 1), and we distinguish some cases.

- If $x \equiv y \equiv 1 \pmod 4$, which only takes $O(1)$ bit operations to check because x and y are given in binary, then x and y both lie in the cyclic subgroup of $(\mathbb{Z}/m_2\mathbb{Z})^*$ generated by the unit 5 (which is equal to the unit 1 if $m_2 = 4$). Therefore, just as for $p > 2$, we have that y is a power of x if and only if $\operatorname{ord}_{m_2}(y) \mid \operatorname{ord}_{m_2}(x)$.
- If $x \equiv 1 \pmod 4$ and $y \equiv 3 \pmod 4$, then y cannot be a power of x modulo m_2.
- If $x \equiv 3 \pmod 4$ and $y \equiv 1 \pmod 4$, then y is a power of x modulo m_2 if and only if y is a power of x^2 modulo m_2. Because $x^2 \equiv 1 \pmod 4$, we conclude that y is a power of x modulo m_2 if and only if $\operatorname{ord}_{m_2}(y) \mid \operatorname{ord}_{m_2}(x^2)$.
- If $x \equiv y \equiv 3 \pmod 4$, then y is a power of x modulo m_2 if and only if $-y$ is a power with odd exponent of $-x$ modulo m_2. Therefore, in order to check whether y is a power of x modulo m_2, we first check whether $\operatorname{ord}_{m_2}(-y) \mid \operatorname{ord}_{m_2}(-x)$. If not, then y is certainly *not* a power of x modulo m_2. Otherwise, we compute $\log_{-x}^{(m_2)}(-y)$ and check whether it is odd.

In summary, since

$$\sum_{p \mid m} \log^k m_p \in O(\log^k m)$$

for all real exponents $k \geq 1$, we conclude that computing $\log_x^{(m_p)}(y)$ for each prime $p \mid m$ takes m-bounded Las Vegas dual complexity $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$ for all p together. As noted above, if any of these "primary discrete logarithms" is ∞, then y is *not* a power of x modulo m, i.e., $\log_x^{(m)}(y) = \infty$, and we are done. Otherwise, noting that for each integer t, we have

$$y \equiv x^t \pmod m \text{ if and only if } y \equiv x^t \pmod{m_p} \text{ for all primes } p \mid m,$$

we find that $\log_x^{(m)}(y) < \infty$ if and only if the system of congruences

$$t \equiv \log_x^{(m_p)}(y) \pmod{\operatorname{ord}_{m_p}(x)} \text{ for all primes } p \mid m$$

in the variable t is consistent, in which case $\log_x^{(m)}(y)$ is its unique solution between 0 and $\operatorname{lcm}(\{\operatorname{ord}_{m_p}(x) : p \mid m\}) - 1 = \operatorname{ord}_m(x) - 1$. We already computed the right-hand sides and moduli of this system of congruences, except possibly $\operatorname{ord}_{m_2}(x)$, which takes m-bounded Las Vegas dual complexity $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$ to compute. Following that, we can check the consistency of this system using the equivalence of statements (1) and (2) in Proposition 2.2.1, which requires $O(\log m)$ subtractions, gcd computations and divisions of integers less than m, hence can be done using $O(\log^{2+o(1)} m)$ bit operations. Should the system be consistent, the integer value of $\log_x^{(m)}(y)$ may be determined by solving the system (which is deterministic and can certainly be done with $O(\log^{7+o(1)} m)$ bit operations, but we do not go into the details of this here), or by computing $\operatorname{ord}_m(x)$ and running the aforementioned routine from [37, p. 144 and Appendix A.2] for another m-bounded Las Vegas dual complexity of $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$. This concludes the proof of statement (3).

For statement (4), we assume that $x, y \in \mathbb{F}_q^*$ are given. In accordance with the two formats (specified at the beginning of this section) in which generalized cyclotomic mappings may be given, we consider two distinct versions of the computational problem of finding $\log_x(y)$:

- version 1: x, y are given as powers of a common, unspecified primitive element ω of $\mathbb{F}_q$;
- version 2: a primitive irreducible polynomial $P(T)$ over $\mathbb{Z}/p\mathbb{Z}$ of degree $\log_p q$ is known, and $\mathbb{F}_q$ is to be viewed as $(\mathbb{Z}/p\mathbb{Z})[T]/(P(T))$, with x, y given as elements of this quotient ring in standard form (i.e., as polynomials over $\mathbb{Z}/p\mathbb{Z}$ in the variable T and of degree less than $\log_p q$).

In either scenario, we show that one can check whether y is a power of x and, if so, work out the integer value of $\log_x(y)$ using q-bounded Las Vegas dual complexity $(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q)$ altogether. Indeed, we note that for any given primitive element $\omega \in \mathbb{F}_q$, the function $\log_\omega : \mathbb{F}_q^* \to \mathbb{Z}/(q-1)\mathbb{Z}$ is a group isomorphism. In the first version of the computational problem, an (unspecified) value for ω was already fixed, and in the second version, we set $\omega := T$ (viewed as an element of $(\mathbb{Z}/p\mathbb{Z})[T]/(P(T))$). In either case, we can work out $\log_\omega(x)$ and $\log_\omega(y)$; in the first scenario, x and y are literally given as powers of ω, and in the second scenario, we apply the routine from [37, p. 144 and Appendix A.2] to compute $\log_\omega(x)$ and $\log_\omega(y)$, which works because x and y are powers of ω and we know that $\operatorname{ord}(\omega) = q - 1$. In either case, those two discrete logarithms can be computed with q-bounded Las Vegas dual complexity $(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q)$ (in fact, in

the first scenario, they only need to be copied from the input, requiring $O(\log q)$ bit operations only).

After finding $\log_\omega(x)$ and $\log_\omega(y)$, the problem is reduced to checking whether $\log_\omega(y)$ is a multiple of $\log_\omega(x)$ in $\mathbb{Z}/(q-1)\mathbb{Z}$, i.e., whether

$$\log_\omega(y) \equiv k \cdot \log_\omega(x) \pmod{q-1} \tag{5.1}$$

for some $k \in \mathbb{Z}$. But this is the case if and only if $\gcd(\log_\omega(x), q-1) \mid \log_\omega(y)$, which can be checked using $O(\log^{1+o(1)} q)$ bit operations. If so, then $\log_x(y)$ is the unique solution $k \in \{0, 1, \ldots, q-2\}$ of congruence (5.1), that is, modulo $q-1$

$$\log_x(y) = \frac{\log_\omega(y)}{\gcd(\log_\omega(x), q-1)} \cdot \mathrm{inv}_{\frac{q-1}{\gcd(\log_\omega(x), q-1)}} \left(\frac{\log_\omega(x)}{\gcd(\log_\omega(x), q-1)} \right),$$

which can be evaluated with a final batch of $O(\log^{1+o(1)} q)$ bit operations. This concludes the proof of statement (4).

For statement (5), we first aim to find a primitive root modulo p. A polynomial-time probabilistic classical algorithm for doing so is [22, Algorithm 1], which is a refinement of an earlier algorithm by Bach [10], itself based on Itoh's idea of using partial factorizations of $p-1$ to find a primitive root with high probability. We follow this approach, but since we can factor $p-1$ completely, our situation is easier. We start by factoring $p-1$, taking p-bounded Las Vegas dual complexity $(\log^{7+o(1)} p, \log^{3+o(1)} p, \log p)$ by statement (1). Say $p - 1 = \prod_{j=1}^{K} p_j^{v_j}$ is the said factorization. For each prime divisor p_j of $p-1$, we wish to find a unit $y_j \in (\mathbb{Z}/p\mathbb{Z})^* = \{1, 2, \ldots, p-1\}$ such that $p_j^{v_j}$ divides $\mathrm{ord}_p(y_j)$. Equivalently, y_j should *not* be a p_j-th power in $\mathbb{Z}/p\mathbb{Z}$. The proportion of units that satisfy this is $1 - \frac{1}{p_j} \geq \frac{1}{2}$, so if we pick $y_j \in (\mathbb{Z}/p\mathbb{Z})^*$ at random, then check whether $y_j^{(p-1)/p_j} \not\equiv 1 \pmod{p}$, it only takes an expected number of $O(1)$ tries until we succeed at finding y_j. As in the proof of statement (1), we perform this random drawing of y_j using a $(\lfloor \log_2(p-2) \rfloor + 1)$-qubit Hadamard circuit. This means that the expected p-bounded Las Vegas dual complexity of finding y_j is $(\log^{2+o(1)} p, \log p, 1)$ for a single j, and $(\log^{3+o(1)} p, \log^2 p, \log p)$ for all $j = 1, 2, \ldots, K$ together. Once the y_j have been found, the unit

$$\mathrm{r} := \prod_{j=1}^{K} y_j^{(p-1)/p_j^{v_j}} \in (\mathbb{Z}/p\mathbb{Z})^*,$$

which can be computed with an additional $O(\log^{3+o(1)} p)$ bit operations, is a primitive root modulo p. Indeed, the j-th factor in this product has order $p_j^{v_j}$, and because the numbers $p_j^{v_j}$ are pairwise coprime and the group $(\mathbb{Z}/p\mathbb{Z})^*$ is abelian, this entails that $\mathrm{ord}_p(\mathrm{r}) = \prod_{j=1}^{K} p_j^{v_j} = p - 1$.

From r, it is not difficult to construct a primitive root r^+ modulo p^k. Indeed, if $k = 1$, we just set $r^+ := r$. Moreover, a primitive root modulo p^2 is a primitive root modulo p^k for each $k \geq 2$, and either r or $r + p$ is a primitive root modulo p^2. Hence, if $k > 1$, we simply check whether $r^{p-1} \equiv 1 \pmod{p^2}$, taking $O(\log^{2+o(1)} p^2) = O(\log^{2+o(1)} p)$ bit operations. If so, we set $r^+ := r + p$, otherwise we set $r^+ := r$. This concludes the proof of statement (5) and of Lemma 5.1.6 as a whole. ∎

Our first application of Lemma 5.1.6 is the following aforementioned result on converting a query complexity into a Las Vegas dual complexity.

Lemma 5.1.7. *Let $\mathfrak{L}$ be an algorithmic problem, and let $y, y_1, \ldots, y_n$ be non-negative real parameters associated with the admissible inputs for $\mathfrak{L}$ such that*

$$y(\vec{x}) \leq h(y_1(\vec{x}), \ldots, y_n(\vec{x}))$$

for all $\vec{x} \in \mathfrak{L}_{\text{in}}$, where h is a fixed function $[0, \infty)^n \to [0, \infty)$. Moreover, let

$$\vec{\mathcal{C}}^{\text{qry}} = (\mathcal{C}_{\text{class}}, \mathcal{C}_{\text{fdl}}, \mathcal{C}_{\text{mdl}}, \mathcal{C}_{\text{mord}}, \mathcal{C}_{\text{prt}})$$

be a y-bounded query complexity of $\mathfrak{L}$ with respect to $y_1, \ldots, y_n$. Then

$$\vec{\mathcal{C}} = (\mathcal{C}'_{\text{class}}, \mathcal{C}_{\text{quant}}, \mathcal{C}_{\text{conv}}),$$

with $\mathcal{C}'_{\text{class}}, \mathcal{C}_{\text{quant}}, \mathcal{C}_{\text{conv}} : [0, \infty)^n \to [0, \infty)$ as defined below, is a y-bounded Las Vegas dual complexity of $\mathfrak{L}$. We write $\vec{z}$ shorthand for $(z_1, z_2, \ldots, z_n) \in [0, \infty)^n$.

$$\begin{aligned}
\mathcal{C}'_{\text{class}}(\vec{z}) &:= \mathcal{C}_{\text{class}}(\vec{z}) + \log^{7+o(1)}(h(\vec{z})) \cdot (\mathcal{C}_{\text{mdl}}(\vec{z}) + \mathcal{C}_{\text{mord}}(\vec{z}) + \mathcal{C}_{\text{prt}}(\vec{z})) \\
&\quad + \log^{3+o(1)}(h(\vec{z}))\mathcal{C}_{\text{fdl}}(\vec{z}); \\
\mathcal{C}_{\text{quant}}(\vec{z}) &:= \log^{3+o(1)}(h(\vec{z})) \cdot (\mathcal{C}_{\text{mdl}}(\vec{z}) + \mathcal{C}_{\text{fdl}}(\vec{z}) + \mathcal{C}_{\text{mord}}(\vec{z}) + \mathcal{C}_{\text{prt}}(\vec{z})); \\
\mathcal{C}_{\text{conv}}(\vec{z}) &:= \log(h(\vec{z})) \cdot (\mathcal{C}_{\text{mdl}}(\vec{z}) + \mathcal{C}_{\text{fdl}}(\vec{z}) + \mathcal{C}_{\text{mord}}(\vec{z}) + \mathcal{C}_{\text{prt}}(\vec{z})).
\end{aligned}$$

Proof. This follows easily from Lemma 5.1.6 and the definitions of the involved concepts. For example, when computing $\mathcal{C}'_{\text{class}}(\vec{z})$, we not only have to take into account the bit operations spent outside the special queries, which are represented by the summand $\mathcal{C}_{\text{class}}(\vec{z})$, but also those coming from the queries, using the algorithms discussed in the proofs of Lemma 5.1.6 to fulfill those queries. For example, on input $\vec{x}$, each of the $O(\mathcal{C}_{\text{fdl}}(y_1(\vec{x}), \ldots, y_n(\vec{x})))$ finite field discrete logarithm queries needed in the course of computing an admissible output for $\vec{x}$ is about computing a discrete logarithm in a finite field of size at most $h(y_1(\vec{x}), \ldots, y_n(\vec{x}))$. Therefore, by Lemma 5.1.6 (4), these "fdl queries" together account for

$$O(\mathcal{C}_{\text{fdl}}(y_1(\vec{x}), \ldots, y_n(\vec{x})) \cdot \log^{3+o(1)}(h(y_1(\vec{x}), \ldots, y_n(\vec{x}))))$$

bit operations, whence the inclusion of the summand $\mathcal{C}_{\mathrm{fdl}}(\vec{z}) \cdot \log^{3+o(1)}(h(\vec{z}))$ in the definition of $\mathcal{C}'_{\mathrm{class}}(\vec{z})$. For the other three kinds of queries, one can use Lemma 5.1.6 together with the fact that

$$\sum_{p|m} \log^t (p^{v_p(m)}) \in O(\log^t m)$$

for each positive integer m and each real exponent $t \geq 1$. For example, for a "prt query", we need to find a primitive root modulo $p^{v_p(m)}$ for each odd prime p dividing m. We do so by first factoring m, then applying the algorithm from Lemma 5.1.6 (5). By Lemma 5.1.6 (1,5), the number of bit operations needed in the process is in

$$O\Big(\log^{7+o(1)} m + \sum_{2<p|m} \log^{7+o(1)} p\Big) \subseteq O\Big(\log^{7+o(1)} m + \sum_{p|m} \log^{7+o(1)} \big(p^{v_p(m)}\big)\Big)$$
$$= O(\log^{7+o(1)} m) \subseteq O(\log^{7+o(1)} y(\vec{x})) \subseteq O(\log^{7+o(1)} h(y_1(\vec{x}), \ldots, y_n(\vec{x}))),$$

which also subsumes the cost of computing $p^{v_p(m)}$ from p and $v_p(m)$ for all p by Lemma 5.1.5 (6). The number of elementary quantum gates, respectively of bit-qubit conversions, needed in the process may be dealt with analogously. Moreover, an analogous approach works for "mdl queries" and "mord queries", where one also needs to factor m (see Lemma 5.1.6 (1)) and (due to the upper bound of $y(\vec{x})^2$ on m from Definition 5.1.3 (4,a)) ends up with an argument of $h(y_1(\vec{x}), \ldots, y_n(\vec{x}))^2$ in the logarithm power, but this may be replaced by $h(y_1(\vec{x}), \ldots, y_n(\vec{x}))$ without changing the O-class of the overall expression. ∎

In view of Lemma 5.1.7, we mostly work with query complexities from here on, only converting them to Las Vegas dual complexities in some main results. In order to solve the three algorithmic problems on index d generalized cyclotomic mappings f from the beginning of this section using the theory developed in this memoir, we first need to compute the induced function $\bar{f} : \{0, 1, \ldots, d\} \to \{0, 1, \ldots, d\}$ and, for each $i \in \{0, 1, \ldots, d-1\}$ such that the coefficient a_i in the cyclotomic form (1.1) of f is non-zero, we need to compute the affine map A_i of $\mathbb{Z}/s\mathbb{Z}$ that encodes the restriction $f_{|C_i} : C_i \to C_{\bar{f}(i)}$ under the identification of C_j with $\mathbb{Z}/s\mathbb{Z}$ via the bijection ι_j described in our introduction. Our next goal is to analyze the query complexity of these tasks.

Proposition 5.1.8. *Given f, one can compute the induced function $\bar{f}$ and the associated affine maps A_i with q-bounded query complexity*

$$(d \log^{1+o(1)} q, d, 0, 0, 0).$$

Proof. With regard to $\bar{f}$, we know that $\bar{f}(d) = d$, so only the values $\bar{f}(i)$ for $i \in \{0, 1, \ldots, d-1\}$ need to be computed. These are $O(d)$ cases. By our discussion in

the introduction, we have $\bar{f}(i) = (e_i + r_i i) \bmod d$, where $e_i = \log_\omega(a_i)$, and by our assumptions from the beginning of this section, this discrete logarithm is either directly specified with a_i, or we compute it with a (finite) field discrete logarithm (fdl) query. After computing e_i, it takes another $O(\log^{1+o(1)} q)$ bit operations by Lemma 5.1.5 (1,3) to evaluate $(e_i + r_i i) \bmod d$ and thus compute $\bar{f}(i)$. In total, a q-bounded query complexity of computing $\bar{f}$ is

$$(d \log^{1+o(1)} q, d, 0, 0, 0).$$

Once $\bar{f}$ has been determined, the computation of the A_i is easy; for each of the $O(d)$ values of i in question, we note that $A_i(x) = \alpha_i x + \beta_i$ for all $x \in \mathbb{Z}/s\mathbb{Z}$, where $\alpha_i, \beta_i \in \mathbb{Z}/s\mathbb{Z}$ are constants. Computing A_i just means computing α_i and β_i, and by the discussion in our introduction, we have

$$\alpha_i = r_i, \quad \beta_i = \frac{e_i + r_i i - \bar{f}(i)}{d} \bmod s.$$

We can directly read off r_i from the definition (1.1) of f, and computing β_i takes $O(\log^{1+o(1)} q)$ bit operations by Lemma 5.1.5 (1,3). Therefore, computing all α_i and β_i after $\bar{f}$ has been worked out takes $O(d \log^{1+o(1)} q)$ bit operations, and the result follows. ∎

With regard to Problem 3 from the beginning of this section, we note that solving this problem efficiently provides us with a quick understanding of each given connected component of Γ_f. While it would be more desirable to have an efficient algorithm that achieves a *global* understanding, of the isomorphism types of all connected components of Γ_f, it is not even clear what the output of such an algorithm would look like. In Definition 5.3.2.8, we introduce the concept of a tree necklace list, which is a way to list the isomorphism types of all connected components of Γ_f with their multiplicities. While such a tree necklace list is a compact encoding of the isomorphism type of Γ_f in *some* cases (e.g., the ones considered in Sections 5.3.2 and 5.3.3), it is not clear whether that is always the case; see also the discussion after Remark 5.3.2.9.

The following main result of this chapter provides q-bounded query and Las Vegas dual complexities of the three algorithmic problems from the beginning of this section. We recall that $\mathrm{mpe}(q-1) = \max_{p|q-1} \nu_p(q-1)$ denotes the maximum exponent of a prime in the prime factorization of $q-1$.

Theorem 5.1.9. *The following hold with regard to the three algorithmic problems from the beginning of this section.*

(1) *Problem 1 has q-bounded query complexity*

$$(d \log^2 d + d^2 \log^{1+o(1)} q + d \log^{2+o(1)} q + \log^{3+o(1)} q, d, d \log q, d, 1)$$

and q-bounded Las Vegas dual complexity

$$(d\log^2 d + d^2\log^{1+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q).$$

(2) *Problem 2 has q-bounded query complexity*

$$(d^3\,\mathrm{mpe}(q-1)2^{(3d^2+d)\,\mathrm{mpe}(q-1)+2d}\log^{1+o(1)} q, d, 0, 0, 0)$$

and q-bounded Las Vegas dual complexity

$$(d^3\,\mathrm{mpe}(q-1)2^{(3d^2+d)\,\mathrm{mpe}(q-1)+2d}\log^{1+o(1)} q + d\log^{3+o(1)} q,$$
$$d\log^{3+o(1)} q, d\log q).$$

Moreover, the computed tree-partition register can be chosen such that its underlying recursive tree description list is of length in

$$O(\min\{d2^{d^2\,\mathrm{mpe}(q-1)+d}, q\}),$$

with each tree description from the list being itself a list of length in

$$O(\min\{d2^{d^2\,\mathrm{mpe}(q-1)+d}, q\}),$$

each entry of which is an ordered pair of bit length in $O(\log q)$.

(3) *Problem 3 has q-bounded query complexity*

$$(d\log^{3+o(1)} q + d^3\,\mathrm{mpe}(q-1)\log^{1+o(1)} q$$
$$+ d^3\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q, d, d^3\,\mathrm{mpe}(q-1), d^3\,\mathrm{mpe}(q-1), 0)$$

and q-bounded Las Vegas dual complexity

$$(d^3\,\mathrm{mpe}(q-1)\log^{7+o(1)} q + d^3\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q,$$
$$d^3\,\mathrm{mpe}(q-1)\log^{3+o(1)} q, d^3\,\mathrm{mpe}(q-1)\log q).$$

We prove Theorem 5.1.9 in Section 5.2. Before that, we make some more comments on the complexities of the three algorithmic problems in Theorem 5.1.9. Firstly, while we are mostly focused on quantum complexities in this memoir, we note that knowing the query complexity of an algorithmic problem also allows one to deduce a bound on its classical Las Vegas complexity. Indeed, all four kinds of queries we are concerned with admit rigorously subexponential solution algorithms on classical computers: for integer factorization, and thus (following the proof of Lemma 5.1.6 (2)) modular multiplicative orders, as well as discrete logarithms in finite fields, see Pomerance's paper [60]. For modular discrete logarithms, see Adleman's

extended abstract [1]. Finally, primitive roots may be dealt with using Dubrois–Dumas' algorithm from [22], combined with modular multiplicative order computations to confirm the alleged primitive root produced by the Dubrois–Dumas algorithm. The upshot of this is that as long as all components of one the q-bounded query complexities in Theorem 5.1.9 are subexponential in the input size (which lies in $O(d \log q)$), then the corresponding problem admits a rigorously subexponential classical Las Vegas solution algorithm.

Secondly, considering the said components of the q-bounded query complexities in Theorem 5.1.9, a glaring question is how the inclusion of the parameter $\mathrm{mpe}(q-1)$ in some components in statements (2) and (3) affects their size. At first glance, this seems rather bad, because generally $\mathrm{mpe}(q-1) \leq \lfloor \log_2(q-1) \rfloor \sim \log_2 q$, and this bound is attained whenever q is a Fermat prime. Since in statements (2) and (3) of Theorem 5.1.9, $\mathrm{mpe}(q-1)$ occurs in the exponent of a power with base 2, this means that in the worst case, the given complexities for Problems 2 and 3 are exponential in the input length for fixed d.

That being said, it turns out that "most of the time", $\mathrm{mpe}(q-1)$ is actually bounded from above by a suitably large constant, as the following result states. This result and its proof was kindly pointed out by MathOverflow user "Dr. Pi" in a response to a question posted by the first author on MathOverflow[1].

Proposition 5.1.10. *There is an absolute constant $c_{\mathrm{mpe}} > 0$ such that for all $x \geq 2$, one has*

$$\Big(\sum_{q \leq x} 1\Big)^{-1} \sum_{q \leq x} \mathrm{mpe}(q-1) \leq c_{\mathrm{mpe}},$$

where the variable q ranges over prime powers. In particular, the following hold.

(1) *For each $\varepsilon > 0$, there is a constant $c_\varepsilon > 0$ such that for all prime powers q except an asymptotic fraction of less than ε, one has $\mathrm{mpe}(q-1) < c_\varepsilon$.*

(2) *Let $h : [0,\infty) \to [0,\infty)$ be a function such that $h(x) \to \infty$ as $x \to \infty$. Then for asymptotically almost all prime powers q, one has $\mathrm{mpe}(q-1) \leq h(q)$.*

Proof. We start by observing that the number of proper (i.e., non-prime) prime powers up to x is asymptotically equivalent to $2x^{1/2}/\log x$. Indeed, the number of prime *squares* up to x is

$$\pi(x^{1/2}) \sim \frac{x^{1/2}}{\log(x^{1/2})} = \frac{2x^{1/2}}{\log x}.$$

Moreover, a prime power $p^k \leq x$ with $k \geq 3$ satisfies $k \leq \lfloor \log_2 x \rfloor$, and for each fixed k, the number of such prime powers is at most $x^{1/k} \leq x^{1/3}$. Hence the number

[1] see https://mathoverflow.net/questions/436134/average-value-of-the-prime-omega-function-omega-on-predecessors-of-prime-powe, visited on 2 September 2025.

of all proper prime powers up to x is

$$\pi(x^{1/2}) + O(x^{1/3}\log x) \sim \frac{2x^{1/2}}{\log x}.$$

This entails the following two things.

(1) The number $\sum_{q\le x} 1$ of *all* prime powers up to x is asymptotically equivalent to $x/\log x$, same as $\pi(x)$.

(2) In the sum $\sum_{q\le x} \mathrm{mpe}(q-1)$, the total contribution stemming from *proper* prime powers is at most

$$O\Big(\log x \cdot \frac{2x^{1/2}}{\log x}\Big) = O(x^{1/2}) \subseteq o\Big(\sum_{q\le x} 1\Big).$$

We may thus focus on the contribution $\sum_{p\le x} \mathrm{mpe}(p-1)$ stemming from primes.

For each $v = 1, 2, \dots, \lfloor \log_2 x \rfloor$, we give a O-bound on the number of primes $p \le x$ with $\mathrm{mpe}(p-1) = v$. For $v = 1$, we use the trivial bound

$$O(\pi(x)) = O\Big(\frac{x}{\log x}\Big) = O\Big(\frac{x}{2\log x}\Big).$$

Now we assume that $v \ge 2$. In order to derive a bound for such v, we use the Brun–Titchmarsh theorem in its stronger form proved by Montgomery and Vaughan [55, Theorem 2]. This result states that for $\mathfrak{a} \in \mathbb{N}^+$, $\mathfrak{b} \in \mathbb{Z}$ and each real $x > \mathfrak{a}$, the number of primes $p \le x$ with $p \equiv \mathfrak{a} \pmod{\mathfrak{b}}$ is at most

$$\frac{2x}{\phi(\mathfrak{b})\log(x/\mathfrak{b})}.$$

Now, a prime $p \le x$ with $\mathrm{mpe}(p-1) = v$ is congruent to 1 modulo $\mathfrak{p}^v$ for some prime $\mathfrak{p} < x^{1/v}$. If $\mathfrak{p}^v \le x^{1/2}$ is fixed, then the Brun–Titchmarsh theorem implies that the number of primes $p \le x$ with $p \equiv 1 \pmod{\mathfrak{p}^v}$ is at most

$$\frac{2x}{\phi(\mathfrak{p}^v)\log(x/\mathfrak{p}^v)} \le \frac{4x}{\mathfrak{p}^{v-1}(\mathfrak{p}-1)\log x} \in O\Big(\frac{x}{\mathfrak{p}^v \log x}\Big).$$

On the other hand, if $x^{1/2} < \mathfrak{p}^v < x$, then the number of primes $p \le x$ with $p \equiv 1 \pmod{\mathfrak{p}^v}$ is at most $x/\mathfrak{p}^v < x^{1/2}$. It follows that the number of primes $p \le x$ with $\mathrm{mpe}(p-1) = v = 2$ is in

$$O\Big(\frac{x}{\log x} \cdot \sum_{\mathfrak{p}\le x^{1/4}} \frac{4}{\mathfrak{p}(\mathfrak{p}-1)} + x^{1/2} \cdot \frac{2x^{1/2}}{\log x}\Big) = O\Big(\frac{x}{\log x}\Big) = O\Big(\frac{x}{4\log x}\Big)$$

and, if $v > 2$, that number is in

$$O\Big(\frac{x}{\log x}\cdot\sum_{\mathfrak{p}\le x^{1/(2v)}}\frac{4}{\mathfrak{p}^{v-1}(\mathfrak{p}-1)}+x^{1/2}\cdot x^{1/3}\Big)=O\Big(\frac{x}{2^v\log x}\Big).$$

In summary, we have shown that for each $v = 1, 2, \ldots, \lfloor \log_2 x \rfloor$, the number of primes $p \le x$ with $\mathrm{mpe}(p-1) = v$ is in $O(x/(2^v \log x))$, and so

$$\sum_{p\le x}\mathrm{mpe}(p-1)\in O\Big(\sum_{v=1}^{\lfloor\log_2 x\rfloor}\frac{vx}{2^v\log x}\Big)=O\Big(\frac{x}{\log x}\sum_{v=1}^{\lfloor\log_2 x\rfloor}\frac{v}{2^v}\Big)=O\Big(\frac{x}{\log x}\Big),$$

whence

$$\Big(\sum_{q\le x}1\Big)^{-1}\sum_{q\le x}\mathrm{mpe}(q-1)\sim\Big(\sum_{p\le x}1\Big)^{-1}\sum_{p\le x}\mathrm{mpe}(p-1)$$
$$\in O\Big(\frac{1}{x/\log x}\cdot\frac{x}{\log x}\Big)=O(1),$$

which is the main statement of this proposition. The first "In particular" statement follows readily from this by observing that the quantity $(\sum_{q\le x}1)^{-1}\sum_{q\le x}\mathrm{mpe}(q-1)$ is the average value of $\mathrm{mpe}(q-1)$ on prime powers $q \le x$. Finally, the second "In particular" statement is an easy consequence of the first. ∎

For applications, finite fields of characteristic 2 are of particular interest. The authors are not aware of any rigorous results concerning the asymptotic behavior of $\mathrm{mpe}(2^v - 1)$ as $v \to \infty$, but in Table 5.1, we provide an overview of the maximum and average values of $\mathrm{mpe}(2^v - 1)$ for $v \in \{1, 2, \ldots, K\}$, where $K \in \{100, 200, \ldots, 1000\}$. This was obtained using GAP [70] and information from the Cunningham project [77]. More specifically, GAP appeared to have difficulties factoring $2^v - 1$ for

$$v \in \{929, 947, 991\},$$

but a quick consultation of the Cunningham factorization tables reveals that

$$\mathrm{mpe}(2^v - 1) = 1$$

for each of these three values of v.

Based on this, we conjecture that the average value of $\mathrm{mpe}(2^v - 1)$ for $1 \le v \le x$ is always less than 2, see Conjecture 6.1.1.

5.2 Proof of Theorem 5.1.9

We give detailed descriptions of algorithms for solving Problems 1–3 and analyze their query complexities (their Las Vegas dual complexities specified in Theorem 5.1.9

K	$\max\{\mathrm{mpe}(2^v-1) : 1 \le v \le K\}$	$K^{-1}\sum_{v=1}^{K} \mathrm{mpe}(2^v-1)$ rounded
100	4	1.28
200	5	1.325
300	5	1.3267
400	5	1.3325
500	6	1.336
600	6	1.3383
700	6	1.3371
800	6	1.34
900	6	1.3389
1000	6	1.341

Table 5.1. Maximum and average values of $\mathrm{mpe}(2^v-1)$.

follow readily using Lemma 5.1.7). The amount of details we give should make it easy to implement these algorithms. In all three cases, we first need to compute $\bar{f}$ and the affine maps A_i, which takes query complexity $(d \log^{1+o(1)} q, d, 0, 0, 0)$ by Proposition 5.1.8. We assume that this has already been done at the start of the discussion of each individual problem. Whenever a positive integer needs to be factored, we subsume this under an mdl or mord query (counted in the third, respectively, fourth, entry of a query complexity). In doing so, we prefer the former but will use the latter in situations where that is more appropriate due to us also needing to compute the multiplicative orders provided by a mord query (this is the case, for example, in the paragraph on $\kappa_{i,p}$ and other parameters in the following section and at the start of the proof of Proposition 5.3.2.3).

5.2.1 Proof of statement (1)

Quite a lot of notations are needed to provide this algorithm in full detail. For the reader's convenience, we print the names of those notations that are newly introduced in this discussion, as well as those of a few notations introduced earlier but rarely used since, in underlined form at the beginning of the respective paragraph where they first appear in this discussion. For the reading flow, these underlined parts need to be ignored. Of course, these notations are also catalogued in Table A.2 in the appendix.

Computing $\bar{\mathcal{L}}$. Before computing $\mathcal{L}$ properly, we need to compute a CRL-list $\bar{\mathcal{L}}$ for $\bar{f}$. Because $\bar{f}$ can be any function $\{0, 1, \dots, d\} \to \{0, 1, \dots, d\}$ with $\bar{f}(d) = d$, we use a general, brute-force algorithm of polynomial complexity in d for this, assuming that the indices $i \in \{0, 1, \dots, d\}$ are processed as non-negative integers in binary representation, with $\lfloor \log_2 d \rfloor + 1$ digits each. Specifically, the following algorithm

based on the idea of "burning leaves", which was kindly pointed out to the authors by one of the reviewers, achieves the computation of $\bar{\mathfrak{L}}$ using $O(d \log^2 d)$ bit operations.

We start by computing three length $d+1$ lists L_1, L_2 and L_3 where, for $j \in \{0, 1, \ldots, d\}$, the $(j+1)$-th entry $L_{1,j}$ of L_1 if $\bar{f}(j)$, and $L_{2,j} := |\bar{f}^{-1}(j)|$ and $L_{3,j} := 0$. Since the values of $\bar{f}$ have already been computed, filling L_1 with its entries is a mere copying process taking $O(d \log d)$ bit operations, and the entries of L_2 can be computed by first letting $L_{2,j} := 0$ for each j, followed by going through the entries of L_1 once, incrementing $L_{2,L_{1,j}}$ for $j = 0, 1, \ldots, d$. This computation of L_2 also takes $O(d \log d)$ bit operations, and so does the (trivial) computation of L_3 (note that we still need to print $O(\log d)$ bits for each 0 entry due to the uniform length of the bit representations).

Now, for a finite functional graph Γ, let $\beta(\Gamma)$ be the (functional) graph obtained from Γ by removing ("burning") all of its leaves. Denoting by $\bar{H}$ the maximum tree height in $\Gamma_{\bar{f}}$, we find that $\Gamma_{\bar{f}} = \beta^0(\Gamma_{\bar{f}}) \supsetneq \beta^1(\Gamma_{\bar{f}}) \supsetneq \cdots \supsetneq \beta^{\bar{H}}(\Gamma_{\bar{f}}) = \beta^{\bar{H}+1}(\Gamma_{\bar{f}})$ and $\beta^{\bar{H}}(\Gamma_{\bar{f}})$ is the subgraph of $\Gamma_{\bar{f}}$ spanned by the periodic vertices. For $h = 0, 1, \ldots, \bar{H}-1$, the vertices in $\mathrm{V}(\beta^h(\Gamma_{\bar{f}})) \setminus \mathrm{V}(\beta^{h+1}(\Gamma_{\bar{f}}))$ are called *new leaves after h burning steps*. For $h = 0$, those are simply the leaves of $\Gamma_{\bar{f}}$. We note that one can also write the set $\mathrm{V}(\beta^h(\Gamma_{\bar{f}})) \setminus \mathrm{V}(\beta^{h+1}(\Gamma_{\bar{f}}))$ as $\mathrm{im}(\bar{f}^h) \setminus \mathrm{im}(\bar{f}^{h+1})$; henceforth, we will denote this set by Layer_h for short.

The aforementioned "burning leaves" algorithm consists of computing, for each $h \in \{0, 1, \ldots, \bar{H}-1\}$, lists L'_h, L''_h and L'''_h of variable length such that L'_h is a repetition-free list of the new leaves after h burning steps (i.e., a list-representation of the set Layer_h), L''_h is a repetition-free list of the $\bar{f}$-images of vertices in L'_h, and L'''_h is a list of the same length as L''_h satisfying

$$L'''_{h,j} = |\bar{f}^{-1}(L''_{h,j}) \cap \mathrm{Layer}_h| \quad \text{for each } j.$$

Throughout this process, we will also update the list L_3 such that at the end of the step for computing L'_h, L''_h and L'''_h, each entry $L_{3,j}$ is the intersection of $\bar{f}^{-1}(j)$ with the set of vertices that are "old" leaves (i.e., elements of Layer_t for some $t < h$).

First, we describe how to obtain L'_0, L''_0 and L'''_0. Go through L_2 once; the zero entries correspond bijectively to the leaves of $\Gamma_{\bar{f}}$, and we can store them repetition-freely in L'_0 using $O(d \log d)$ bit operations. Then store the $\bar{f}$-images of the vertices in L'_0 in the list L''_0 – first with repetitions, then sort and remove repetitions in L''_0, storing the repetition multiplicities in L'''_0. By Lemma 5.1.5 (10), this process takes $O(|L'_0| \log^2 |L'_0|)$ bit operations overall. We may leave all entries of L_3 as 0 for now, since there are no old leaves yet after 0 burning steps.

Now, assume that for a given $h \in \{0, 1, \ldots, \bar{H}-2\}$, we have already computed L'_h, L''_h and L'''_h as well as adjusted L_3 such that it counts the number of old leaf pre-images after h burning steps for each vertex. We would like to compute L'_{h+1},

L''_{h+1} and L'''_{h+1} as well as readjust L_3, so it includes the "new leaves after h burning steps" as old leaves.

First, we describe how to compute L'_{h+1}, and we note that this part of the computations will also need to be done for $h = \bar{H} - 1$, as it is through the equality $L_{\bar{H}} = \emptyset$ that we know we are done (i.e., all transient vertices have been burned as leaves at some point and we have reduced the graph to its periodic vertices).

We go through the entries $L''_{h,j}$ of L''_h, and for each of them, we check whether $L'''_{h,j} + L_{3,L''_{h,j}} = L_{2,j}$. If so, this means that the pre-image $\bar{f}^{-1}(L''_{h,j})$ consists entirely of new leaves after h burning steps and older leaves, whence $L''_{h,j}$ will be a new leaf after $h+1$ burning steps, and we may thus store $L''_{h,j}$ in L'_{h+1}. If not, there is some vertex in the pre-image $\bar{f}^{-1}(L''_{h,j})$ that has never been a leaf so far and will thus "survive" the $(h+1)$-th burning step, whence $L''_{h,j}$ will *not* become a new leave after that burning step. In both cases, we update $L_{3,L''_{h,j}}$ by adding $L'''_{h,j}$ to it to account for the newly burned leaves in step $h+1$. This process of computing the new list L'_{h+1} and updating L_3 takes $O(|L''_h| \log d) \subseteq O(|L'_h| \log d)$ bit operations.

Following that (assuming that $h < \bar{H} - 1$), we may compute L''_{h+1} and L'''_{h+1} based on L'_{h+1} analogously to how we computed L''_0 and L'''_0 based on L'_0 above; this takes $O(|L'_h| \log^2 d)$ bit operations.

Overall, the process of computing the lists L'_h, L''_h and L'''_h for each h takes

$$O(d \log d) + \sum_{i=0}^{\bar{H}-2} O(|\mathrm{Layer}_h| \log^2 d) + O(|\mathrm{Layer}_{\bar{H}-1}| \log d)$$
$$\subseteq O(d \log d) + O\left(\sum_{i=0}^{\bar{H}-2} |\mathrm{Layer}_h| \log^2 d \right) = O(d \log^2 d)$$

bit operations, where the last equality uses that the sets Layer_h are pairwise disjoint.

Finally, we can compute the set of periodic points $\mathrm{per}(\bar{f})$ as the complement of $\bigcup_{h=0}^{\bar{H}-1} \mathrm{Layer}_h$ using another $O(d \log d)$ bit operations. Then we determine $\bar{\mathcal{L}}$ through iteration of $\bar{f}$ on $\mathrm{per}(\bar{f})$ by brute force. If we mark vertices as visited and keep track of the already processed initial segment of the list representing $\mathrm{per}(\bar{f})$ internally, we can do so using $O(d \log d)$ bit operations only, and in the process, we can actually store each cycle of $\bar{f}$ in full, which will be useful shortly. In total, the computations for $\bar{\mathcal{L}}$ require $O(d \log^2 d)$ bit operations.

Computing $\mathcal{L}_i$, U_i, par_i, Y_i. To compute the desired parametrization of $\mathcal{L}$, we go through the elements $(i, \ell) \in \bar{\mathcal{L}}$, with associated $\bar{f}$-cycle $(i_0, i_1, \ldots, i_{\ell-1})$, and compute a parametrization of a CRL-list $\mathcal{L}_i$ of the restriction $f_{|U_i}$, where $U_i = \bigcup_{t=0}^{\ell-1} C_{i_t}$. This works because by Proposition 3.1.1, $\mathcal{L}$ is simply the (disjoint) union of those $\mathcal{L}_i$. Specifically, we compute a formula that defines a bijective function $\mathrm{par}_i : Y_i \to \mathcal{L}_i$, where Y_i is a "simple" set depending on i. For $(i, \ell) = (d, 1)$, which is dealt with

outside the loop for the other pairs (i,ℓ), we have $\mathfrak{L}_d = \{(0_{\mathbb{F}_q}, 1)\}$, and we set $Y_d := \{(\emptyset,\emptyset)\}$ (to conform with the format the sets Y_i for $i < d$ have – each of the two $\emptyset$ is to be viewed as an empty tuple) and define

$$\mathrm{par}_d(\emptyset,\emptyset) := (0_{\mathbb{F}_q}, 1).$$

This only takes $O(\log d)$ bit operations (not $O(1)$, because the index d on the left-hand side of the definition needs to be spelled out).

Computing r_p, $\mathcal{A}_i$, $\overline{\alpha}_i$, $\overline{\beta}_i$, par'_i, $\mathfrak{L}'_i$, $\mathfrak{P}_i$. Next, we factor $s = (q-1)/d$ in a single q-bounded modular discrete logarithm (mdl) query (we remind the reader that we subsume factorizations under mdl queries). We also find a primitive root r_p modulo $p^{\nu_p(s)}$ for each odd prime divisor p of s using a single q-bounded primitive root (prt) query. Following that, we loop over the elements $(i,\ell) \in \overline{\mathfrak{L}}$ with $i < d$, and for each of them, we do the following. We compute

$$\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}, \quad \mathcal{A}_i(z) = \overline{\alpha}_i z + \overline{\beta}_i.$$

This takes $O(d)$ multiplications of already computed affine maps, each of which costs $O(\log^{1+o(1)} q)$ bit operations by Lemma 5.1.5 (1,3) using the formula

$$(z \mapsto \alpha z + \beta)(z \mapsto \alpha' z + \beta') = (z \mapsto \alpha\alpha' z + \alpha'\beta + \beta').$$

Hence, in total, the computation of $\mathcal{A}_i$ takes $O(d \log^{1+o(1)} q)$ bit operations. Our next goal is to compute a parametrization $\mathrm{par}'_i : Y_i \to \mathfrak{L}'_i$ of a CRL-list $\mathfrak{L}'_i$ for $\mathcal{A}_i$, from which $\mathrm{par}_i : Y_i \to \mathfrak{L}_i$ is obtained simply by stretching all second entries (cycle lengths) of images of par'_i by the factor ℓ. As preparations for an upcoming loop over the prime divisors of s, we initialize $\mathfrak{P}_i := \emptyset$ (ultimately, $\mathfrak{P}_i$ will be a list of those prime divisors of s that do *not* divide $\overline{\alpha}_i$). We also compute $\mathrm{ord}_{p^{\nu_p(s)}}(\overline{\alpha}_i)$ for each prime divisor p of s, requiring a single q-bounded multiplicative order (mord) query.

Computing κ_p, $\overline{\mathcal{A}}_{i,p}$, $\mathrm{par}'_{i,p}$, $Y_{i,p}$, $\mathfrak{L}'_{i,p}$. Next, we loop over the prime divisors p of s, and for each of them, we do the following. First, we check whether $p \mid \overline{\alpha}_i$, and if so, we skip to the next value of p. Otherwise, we add p to $\mathfrak{P}_i$, then read off $\kappa_p := \nu_p(s)$ and p^{κ_p} from the factorization of s computed earlier. Following that, we compute $\overline{\mathcal{A}}_{i,p} := \mathcal{A}_i \bmod p^{\kappa_p}$ (that is, we compute $\overline{\alpha}_i \bmod p^{\kappa_p}$ and $\overline{\beta}_i \bmod p^{\kappa_p}$), which takes $O(\log^{1+o(1)} q)$ bit operations by Lemma 5.1.5 (3). We note that since p does not divide $\overline{\alpha}_i$, the function $\overline{\mathcal{A}}_{i,p}$ is an affine *permutation* of $\mathbb{Z}/p^{\kappa_p}\mathbb{Z}$, and from our Table 2.2, we can read off a compact parametrization $\mathrm{par}'_{i,p} : Y_{i,p} \to \mathfrak{L}'_{i,p}$ of a CRL-list $\mathfrak{L}'_{i,p}$ of $\overline{\mathcal{A}}_{i,p}$ in which all specified cycle lengths are fully factored. The details of this are given in Table 5.2 below; each numbered row of that table corresponds to the case with the same number in Table 2.2. The following paragraph introduces some more notation, which is used in Table 2.2 and needs to be computed before one is able to print a description of $\mathrm{par}'_{i,p}$.

No.	u	u'	$\mathrm{par}'_{i,p}(\vec{u})$
1	$0,\dots,\kappa_p$	$0,\dots,$ $\big(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}-v'_{i,p,k}}\big)^{\delta_{[u<\kappa_p]}}\cdot$ $p^{\min\{\kappa_p-1-v'_{i,p},\delta_{[u<\kappa_p]}(\kappa_p-u-1)\}}$ -1	$(\mathfrak{r}_p^{u'}\,p^u+\mathfrak{f}_{i,p},$ $\big(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}}\big)^{\delta_{[u<\kappa_p]}}\cdot$ $p^{\delta_{[u<\kappa_p]}\max\{v'_{i,p}-u,0\}})$
2	$0,\dots,p^{\kappa_{i,p}}-1$	n/a	$(u,\,p^{\kappa_p-\kappa_{i,p}})$
3	$0,\dots,2^{\kappa_{i,2}}-1$	n/a	$(u,2^{\kappa_2-\kappa_{i,2}})$
4	$0,1,2$	n/a	$(u,-u^2+2u+1)$
5	$0,1,2$	n/a	$(2^u-1,\frac{1}{2}u^2-\frac{3}{2}u+2)$
6	$0,1$	n/a	$(2u,2)$
7	$-\kappa_2,\dots,\kappa_2-1$	if $u\in\{-\kappa_2,\kappa_2-1\}$: 0; otherwise: $0,\dots,2^{\kappa_2-2-\max\{v'_{i,2},u\}}-1$	if $u=\kappa_2-1$: $(\mathfrak{f}_{i,2},1)$; if $u=-\kappa_2$: $(2^{\kappa_2-1}+\mathfrak{f}_{i,2},1)$; if $0\le u<\kappa_2-1$: $(5^{u'}2^u+\mathfrak{f}_{i,2},2^{\max\{v'_{i,2}-u,0\}})$; if $-\kappa_2<u<0$: $(-5^{u'}2^{-u-1}+\mathfrak{f}_{i,2},$ $2^{\max\{v'_{i,2}+u+1,0\}})$.
8	$-v''_{i,2},\dots,v''_{i,2}$	if $u=v''_{i,2}$: $0,\dots,2^{\kappa_2-v''_{i,2}-1}$; otherwise: $0,\dots,2^{\kappa_2-v''_{i,2}-2}-1$.	if $u=v''_{i,2}$ and $u'\in\{0,2^{\kappa_2-v''_{i,2}-1}\}$: $(u'2^{v''_{i,2}}+\mathfrak{f}_{i,2},1)$; if $u=v''_{i,2}$ and $0<u'<2^{\kappa_2-v''_{i,2}-1}$: $(u'2^{v''_{i,2}}+\mathfrak{f}_{i,2},2)$; if $0\le u<v''_{i,2}$: $(5^{u'}2^u+\mathfrak{f}_{i,2},2^{v''_{i,2}-u})$; if $u<0$: $(5^{u'}2^{-u-1}+\mathfrak{f}_{i,2},2^{v''_{i,2}+u+1})$.
9	$0,\dots,2^{\kappa_{i,2}}-1$	n/a	$(u,2^{\kappa_2-\kappa_{i,2}})$
10	$1,\dots,2^{\kappa_2-v''_{i,2}-1}$	n/a	if $u=1$: $(0,2^{v''_{i,2}+1})$; otherwise: $(\bar{\beta}_i u,2^{v''_{i,2}+1})$.

Table 5.2. Explicit parametrizations of CRL-lists of affine permutations of finite primary cyclic groups.

Computing $\kappa_{i,p}$, $\mathfrak{n}_p$, $\mathfrak{p}_{p,k}$, $v_{p,k}$, $v'_{i,p,k}$, $v'_{i,p}$, $v''_{i,2}$, $\mathfrak{r}_p$, $\mathfrak{f}_{i,p}$. Recalling that $\nu_p^{(v)}(m):=\min\{\nu_p(m),v\}$, we set

$$\kappa_{i,p}:=\nu_p^{(\kappa_p)}(\bar{\beta}_i)=\nu_p^{(\kappa_p)}(\bar{\beta}_i \bmod p^{\kappa_p}),$$

which can be computed using $O(\kappa_p) \subseteq O(\log p^{\kappa_p})$ integer divisions by p, resulting in a bit operation cost of $O(\log^{2+o(1)} p^{\kappa_p})$. If $p > 2$, we next compute factorizations of $p-1$ and of $\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)$ using $2 \in O(1)$ mord queries. We spell these factorizations out as follows:

$$p - 1 = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}}$$

and

$$\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i) = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}} \cdot p^{v'_{i,p}}.$$

We note that some of the exponents $v'_{i,p,k}$ or $v'_{i,p}$ may be 0. On the other hand, if $p = 2$ (where $\mathfrak{n}_p = 0$), we write

$$\mathrm{ord}_{2^{\kappa_2}}(\overline{\alpha}_i) = 2^{v'_{i,2}},$$

which matches with the notation for $p > 2$ above, and $\mathrm{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i) = 2^{v''_{i,2}}$. Usually, $v''_{i,2} = v'_{i,2}$ as $\mathrm{ord}_{2^{\kappa_2}}(\overline{\alpha}_i) = \mathrm{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i)$, but if $\overline{\alpha}_i \equiv \pm 1 \pmod{2^{\kappa_2}}$, then $v'_{i,2} \in \{0, 1\}$ and $v''_{i,2} = 1 - v_{i,2}$. Finally, regardless of whether or not $p > 2$, we check whether $\overline{\mathcal{A}}_{i,p}$ has a fixed point, i.e., whether

$$\gcd(\overline{\alpha}_i - 1, p^{\kappa_p}) = \gcd((\overline{\alpha}_i \bmod p^{\kappa_p}) - 1, p^{\kappa_p}) \mid \overline{\beta_i} \bmod p^{\kappa_p},$$

which can be done using $O(\log^{1+o(1)} p^{\kappa_p})$ bit operations. We store this information, and whenever $\overline{\mathcal{A}}_{i,p}$ has a fixed point, we compute one, denoted by $\mathfrak{f}_{i,p}$, via the formula in Proposition 2.3.6, taking another $O(\log^{1+o(1)} p^{\kappa_p})$ bit operations.

Computing u, u', $\vec{u}$. We are now ready to give the tabular definition of the bijective parametrization $\mathrm{par}'_{i,p} : Y_{i,p} \to \mathcal{L}'_{i,p}$ of a CRL-list $\mathcal{L}'_{i,p}$ of $\overline{\mathcal{A}}_{i,p}$. We note that the set $Y_{i,p}$ always has one of the following two forms, which will be important later on.

- $Y_{i,p}$ is an integer interval, a general element of which is denoted by u; or
- the elements of $Y_{i,p}$ are pairs (u, u') of integers, where u ranges over an integer interval, and for each fixed value of u, the second entry u' also ranges over an integer interval.

We observe that having entire intervals of integers (or pairs of integers) subsumed under a uniform parametrization like $\mathrm{par}'_{i,p}$ is key to describing the possibly superpolynomially many elements of a CRL-list using just polynomially many bits (the core idea why Problem 1 from the beginning of this chapter can be solved at all).

To have a uniform notation, we may also denote an element of $Y_{i,p}$ by $\vec{u}$ in either of the two cases above. For example, to derive the formulas in the first case of Table 5.2, we apply the first case in Table 2.2, with $v := \kappa_p$, $t := u$, $j := u'$,

$a := \bar{\alpha}_i$ and $b := \bar{\beta}_i$. Then the range for u is clear from the range for t in Table 2.2. Concerning the asserted range for u', we note that

$$\frac{\phi(p^{\kappa_p})}{\mathrm{ord}_{p^{\kappa_p}}(\bar{\alpha}_i)} = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}-v'_{i,p,k}} \cdot p^{\kappa_p-1-v'_{i,p}}$$

and

$$\phi(p^{\kappa_p-u}) = ((p-1)p^{\kappa_p-u-1})^{\delta_{[u<\kappa_p]}} = \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}} \cdot p^{\kappa_p-u-1}\right)^{\delta_{[u<\kappa_p]}},$$

from which it can be deduced that

$$\begin{aligned}&\gcd\left(\frac{\phi(p^{\kappa_p})}{\mathrm{ord}_{p^{\kappa_p}}(\bar{\alpha}_i)}, \phi(p^{\kappa_p-u})\right)\\ &\quad= \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}-v'_{i,p,k}}\right)^{\delta_{[u<\kappa_p]}} \cdot p^{\min\{\kappa_p-1-v'_{i,p},\delta_{[u<\kappa_p]}(\kappa_p-u-1)\}},\end{aligned}$$

as required. Finally, the formula for the cycle lengths (second entries of $\mathrm{par}'_{i,p}(\vec{u})$) in case 1 holds because

$$\begin{aligned}&\frac{\phi(p^{\kappa_p-u})}{\gcd\left(\frac{\phi(p^{\kappa_p})}{\mathrm{ord}_{p^{\kappa_p}}(\bar{\alpha}_i)}, \phi(p^{\kappa_p-u})\right)}\\ &\quad= \frac{\left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}} \cdot p^{\kappa_p-u-1}\right)^{\delta_{[u<\kappa_p]}}}{\left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}-v'_{i,p,k}}\right)^{\delta_{[u<\kappa_p]}} \cdot p^{\min\{\kappa_p-1-v'_{i,p},\delta_{[u<\kappa_p]}(\kappa_p-u-1)\}}}\\ &\quad= \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}}\right)^{\delta_{[u<\kappa_p]}} \cdot p^{\delta_{[u<\kappa_p]}(\kappa_p-u-1-\min\{\kappa_p-1-v'_{i,p},\kappa_p-u-1\})}\\ &\quad= \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}}\right)^{\delta_{[u<\kappa_p]}} \cdot p^{\delta_{[u<\kappa_p]}\max\{v'_{i,p}-u,0\}}.\end{aligned}$$

The other cases in Table 5.2 can be dealt with analogously. To prevent confusion among readers, we note that in cases 7 and 8 of Table 2.2, the specified CRL-list consists of several disjoint parts with different formulas. Because we want u to range over an integer interval, these have been slightly rearranged and "glued together" here. For example, in case 7 here, the ranges $\{0, 1, \ldots, \kappa_2 - 2\}$ and $\{-\kappa_2 + 1, -\kappa_2 + 2, \ldots, -1\}$ for u correspond, respectively, to the parts with representative elements $5^j 2^t + \dagger$ and $-5^j 2^t + \dagger$ in case 7 of Table 2.2. In the latter of the two segments, the range for u is not equal to the corresponding range for t in Table 2.2, which explains the variable substitution $u \to -u - 1$ although the corresponding formulas for cycle lengths in Table 2.2 are the same. Our algorithm prints and stores the parametric description of

$\mathrm{par}'_{i,p}(\vec{u})$ for all primes $p \mid s$ with $p \nmid \overline{\alpha}_i$. For a given p, this parametric description takes $O(\log p^{\kappa_p})$ bits to store (as follows by observing that it takes $O(\log n)$ bits to print the prime factorization of $n \in \mathbb{N}^+$), and so all descriptions together can be stored using $O(\log s) \subseteq O(\log q)$ bits.

Computing par''_i, $\mathcal{L}''_i$, $\mathcal{A}'_i$, s'_i, $\vec{\vec{u}}$, $\vec{u}_p$, u_p, u'_p, $\overline{Y}_i$, proj_j, $\vec{r}_i(\vec{\vec{u}})$, $r_{i,p}(\vec{u}_p)$, $B_{\vec{r}_i(\vec{\vec{u}})}$, $\mathcal{I}_{i,\vec{\vec{u}}}$. Next, using the parametrizations $\mathrm{par}'_{i,p}$ of the CRL-lists $\mathcal{L}'_{i,p}$ of $\overline{\mathcal{A}}_{i,p}$ for $p \in \mathfrak{P}_i$, we construct a parametrization $\mathrm{par}''_i : Y_i \to \mathcal{L}''_i$ of a CRL-list $\mathcal{L}''_i$ for $\mathcal{A}'_i := \mathcal{A}_i \bmod \prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$. We start by setting $s'_i := \prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$, which takes $O(\log^{2+o(1)} q)$ bit operations to compute, carrying out $|\mathfrak{P}_i| \in O(\log q)$ integer multiplications, each with a bit operation cost in $O(\log^{1+o(1)} q)$ (we note that the powers p^{κ_p} themselves do not need to be computed, as they are specified, alongside the pairs (p, κ_p), in the output of the mdl query that gave the factorization of s). We follow the approach described at the end of Section 2.3. More specifically, we identify $\mathbb{Z}/s'_i\mathbb{Z}$ with $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$, and $\mathcal{A}'_i$ with $\bigotimes_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,p}$. We consider tuples $\vec{\vec{u}} = (\vec{u}_p)_{p\in\mathfrak{P}_i} \in \prod_{p\in\mathfrak{P}_i} Y_{i,p} =: \overline{Y}_i$. We can either write $\vec{u}_p = u_p$ or $\vec{u}_p = (u_p, u'_p)$. For $j = 1, 2$, we denote by proj_j the (class-sized) function that maps an ordered pair to its j-th entry. Associated with each parameter tuple $\vec{\vec{u}} \in \overline{Y}_i$, we have the tuple

$$\vec{r}_i(\vec{\vec{u}}) := (r_{i,p}(\vec{u}_p))_{p\in\mathfrak{P}_i} := (\mathrm{proj}_1(\mathrm{par}'_{i,p}(\vec{u}_p)))_{p\in\mathfrak{P}_i}$$

of associated cycle representatives of the $\overline{\mathcal{A}}_{i,p}$. By our discussion at the end of Section 2.3, these tuples $\vec{r}_i(\vec{\vec{u}})$ parametrize the blocks $B_{\vec{r}_i(\vec{\vec{u}})}$ of a certain partition of $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$, each block of which is a union of cycles of $\mathcal{A}'_i$. In terms of $\vec{\vec{u}}$, we wish to explicitly describe a CRL-list for the restriction of $\mathcal{A}'_i$ to $B_{\vec{r}_i(\vec{\vec{u}})}$. For this, we need to exhibit an $\vec{r}_i(\vec{\vec{u}})$-admissible indexing function $\mathcal{I}_{i,\vec{\vec{u}}}$ in the sense of Definition 2.3.8 (1) and understand its associated set of good tuples (in the sense of Definition 2.3.8 (2)).

Computing $l_{i,p,\vec{u}_p}$, $l_{i,\vec{\vec{u}}}$, $\mathfrak{P}'_i$. Now, following the definition of an admissible indexing function, the domain of definition of $\mathcal{I}_{i,\vec{\vec{u}}}$ is the set of all primes that divide at least one of the component cycle lengths $l_{i,p,\vec{u}_p} := \mathrm{proj}_2(\mathrm{par}'_{i,p}(\vec{u}_p))$ for $p \in \mathfrak{P}_i$, or, equivalently, that divide $l_{i,\vec{\vec{u}}} := \mathrm{lcm}\{l_{i,p,\vec{u}_p} : p \in \mathfrak{P}_i\}$, of which we compute a parametric definition of bit length in $O(\log^{1+o(1)} q)$ for later use by scanning the displayed parametric factorizations of the $l_{i,p,\vec{u}_p}$, taking $O(\log^{2+o(1)} q)$ bit operations. By definition, the domain of $\mathcal{I}_{i,\vec{\vec{u}}}$ is a subset of

$$\mathfrak{P}'_i := \mathfrak{P}_i \cup \pi\Big(\prod_{p\in\mathfrak{P}_i} (p-1)\Big).$$

We compute $\mathfrak{P}'_i$ as a list (with $O(\log q)$ entries), and this computation consists of $O(\log q)$ containment checks each involving $O(1)$ copying processes of bit strings

of length in $O(\log q)$, and $O(\log q)$ bit comparisons and scans of memory addresses each of length in $O(\log\log q)$. Hence, we may compute $\mathfrak{P}'_i$ using $O(\log^{2+o(1)} q)$ bit operations. In our algorithmic approach, we treat $\mathcal{I}_{i,\vec{\vec{u}}}$ as a function whose domain of definition is all of $\mathfrak{P}'_i$; the additional primes $\mathfrak{p}$ are those which do not divide any component cycle length, hence occur with valuation 0 in each component, and the value $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$ may be chosen arbitrarily in $\mathfrak{P}_i$. We note that this change does not affect the associated notion of good tuples and ensures that the domain of $\mathcal{I}_{i,\vec{\vec{u}}}$ does not depend on $\vec{\vec{u}}$.

For each $\mathfrak{p} \in \mathfrak{P}'_i$, the value $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$ is a prime $p' \in \mathfrak{P}_i$ (thought of as an index for a component of $\vec{\vec{u}}$) such that $\nu_{\mathfrak{p}}(l_{i,p',\vec{\vec{u}}})$ is maximal among all $\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$ for $p \in \mathfrak{P}_i$. We recall that we have already worked out explicit factorizations of the positive integers $l_{i,p,\vec{u}_p}$ in terms of $\vec{u}_p$ (see Table 5.2). If $\mathfrak{p} \notin \mathfrak{P}_i$, then for each $p \in \mathfrak{P}_i$, the value of $\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$ is constant, not depending on $\vec{u}_p$, and a scan along the length $O(\log q)$ parametric description, combined with comparisons of the relevant exponents $\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$, each of which has bit length in $O(\log\log q)$, lets us pick a suitable value for $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$. For a given $\mathfrak{p} \in \mathfrak{P}'_i \setminus \mathfrak{P}_i$, this process requires $O(\log^{1+o(1)} q)$ bit operations, and carrying it out for all $\mathfrak{p} \in \mathfrak{P}'_i \setminus \mathfrak{P}_i$ takes $O(\log^{2+o(1)} q)$ bit operations.

Computing $\mathcal{J}_{i,p,k}$, $\mathfrak{m}_{i,p}$, $\mathcal{J}'_{i,p,k}$. We still need to discuss the approach when $\mathfrak{p} \in \mathfrak{P}_i$. Even then, $\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$ does not depend on $\vec{u}_p$ unless $p = \mathfrak{p}$, in which case one of the following applies.

- $\nu_p(l_{i,p,\vec{\vec{u}}})$ also does not depend on $\vec{u}_p$ (see, e.g., case 2 in Table 5.2), and we can compute a constant value for $\mathcal{I}_{i,\vec{\vec{u}}}(p)$ as described above.
- $\nu_p(l_{i,p,\vec{\vec{u}}})$ does depend on $\vec{u}_p$, in the following way: it only depends on u_p (not u'_p), and one can partition the range for u_p into at most five subintervals $\mathcal{J}_{i,p,1}, \ldots, \mathcal{J}_{i,p,\mathfrak{m}_{i,p}}$ (case 8 in Table 5.2 does require $\mathfrak{m}_{i,p} = 5$) such that in case $u_p \in \mathcal{J}_{i,p,k}$ for a fixed $k \in \{1, \ldots, \mathfrak{m}_{i,p}\}$, the value of $\nu_p(l_{i,p,\vec{\vec{u}}})$ is either constant, or given by a linear expression in u_p, or given by an expression that is the maximum among a linear expression in u_p and 0. This allows us to specify a subinterval (in fact, an initial or terminal segment) $\mathcal{J}'_{i,p,k}$ of $\mathcal{J}_{i,p,k}$ (with constant boundary points) such that $\mathcal{I}_{i,\vec{\vec{u}}}(p)$ may be chosen as p if $u_p \in \mathcal{J}'_{i,p,k}$, whereas $\mathcal{I}_{i,\vec{\vec{u}}}(p)$ must be chosen as a different constant value in $\mathfrak{P}_i$ (the same for each k) if $u_p \in \mathcal{J}_{i,p,k} \setminus \mathcal{J}'_{i,p,k}$. For each given p, writing down an explicit definition of $\mathcal{I}_{i,\vec{\vec{u}}}(p)$ (which consists of a case distinction with at most two cases) requires us to scan the parametric descriptions of the component images $\mathrm{par}'_{i,p}(\vec{u}_p)$ and perform some low-cost computations such as additions or subtractions between exponents of primes (which are numbers of bit length in $O(\log\log q)$). For all relevant values of $\mathfrak{p}$ together, this can be done using $O(\log^{2+o(1)} q)$ bit operations.

We note that for each given $\mathfrak{p} \in \mathfrak{P}'_i$, the parametric definition of $\mathcal{I}_{i,\vec{\vec{u}}}$ which we just derived has bit length in $O(\log q)$. Therefore, and because the domain $\mathfrak{P}'_i$ of $\mathcal{I}_{i,\vec{\vec{u}}}$

has size in $O(\log q)$, it takes $O(\log^2 q)$ bits to store the parametric definitions of all function values of $\mathcal{I}_{i,\vec{u}}$.

Computing $\mathcal{I}_i$, $\mathfrak{P}_{i,p,\vec{u}}$, $\vec{k}$, k_p, $\mathfrak{d}_{i,p,\vec{u}}$, $\vec{k}'$, k'_p, $K'_{i,\vec{u}}$. Before we proceed with our argument, we need to introduce another notation. For $\mathfrak{p} \in \mathfrak{P}'_i$, if $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ only assumes one distinct value as $\vec{u}$ ranges over $\overline{Y}_i$, we set $\mathcal{I}_i(\mathfrak{p}) := \mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ for any $\vec{u} \in \overline{Y}_i$. On the other hand, if $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ assumes two distinct values, one of which is $\mathfrak{p}$, we let $\mathcal{I}_i(\mathfrak{p})$ be the unique element of $\{\mathcal{I}_{i,\vec{u}}(\mathfrak{p}) : \vec{u} \in \overline{Y}_i\}$ that is distinct from $\mathfrak{p}$. This defines a function $\mathcal{I}_i : \mathfrak{P}'_i \to \mathfrak{P}_i$ that is independent of $\vec{u}$ and can be easily derived from the parametric definitions of the function values $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ (taking $O(\log^{2+o(1)} q)$ bit operations). Using the function $\mathcal{I}_i$, we can give the following compact parametric definition of the pre-image of a singleton subset of $\mathfrak{P}_i$ under $\mathcal{I}_{i,\vec{u}}$:

$$\begin{aligned}\mathfrak{P}_{i,p,\vec{u}} &:= \mathcal{I}_{i,\vec{u}}^{-1}(\{p\}) \\ &= \left(\mathcal{I}_i^{-1}(\{p\}) \setminus \{\mathfrak{p} \in \mathfrak{P}_i \setminus \{p\} : u_{\mathfrak{p}} \in \bigcup_{k=1}^{\mathfrak{m}_{i,\mathfrak{p}}} \mathcal{J}'_{i,p,k}\}\right) \\ &\quad \cup \left\{\mathfrak{p} \in \{p\} : u_{\mathfrak{p}} \in \bigcup_{k=1}^{\mathfrak{m}_{i,\mathfrak{p}}} \mathcal{J}'_{i,p,k}\right\}. \end{aligned} \tag{5.2}$$

We note that our algorithm is merely producing this defining formula for $\mathfrak{P}_{i,p,\vec{u}}$ for each $p \in \mathfrak{P}_i$, which is harmless complexity-wise – even when spelling $\mathcal{I}_i^{-1}(\{p\})$ out explicitly in each case, this can be done using $O(\log^{1+o(1)} q)$ bit operations and storage space per p, hence $O(\log^{2+o(1)} q)$ bit operations and storage space altogether. One could also try to provide a case-distinction definition of $\mathfrak{P}_{i,p,\vec{u}}$, where each case corresponds to a constant value of $\mathfrak{P}_{i,p,\vec{u}}$, but this breaks the complexity, as one needs to go through $2^{O(\log q)}$ cases in general. Likewise, it is easy to check that producing each parametric definition described in the rest of this argument takes $O(\log^{2+o(1)} q)$ bit operations if one is careful enough about how to spell those parametrizations out.

Having these explicit definitions of the pre-images $\mathfrak{P}_{i,p,\vec{u}}$ is important because they are needed to set up a parametrization of the $\mathcal{I}_{i,\vec{u}}$-good tuples. We recall from above the notation $l_{i,p,\vec{u}_p}$ for the cycle length of the representative $r_{i,p}(\vec{u}_p)$ in the p-indexed component of $\vec{r}_i(\vec{u})$. An $\mathcal{I}_{i,\vec{u}}$-good tuple is a tuple $\vec{k} = (k_p)_{p \in \mathfrak{P}_i}$ with $k_p \in \mathbb{Z}/l_{i,p,\vec{u}_p}\mathbb{Z} = \{0, 1, \ldots, l_{i,p,\vec{u}_p} - 1\}$ such that k_p is divisible by

$$\mathfrak{d}_{i,p,\vec{u}} := \prod_{\mathfrak{p} \in \mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,\vec{u}})}.$$

We can compute a parametric definition of $\mathfrak{d}_{i,p,\vec{u}}$ and $l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{u}}$ using storage space per p and $O(\log^{1+o(1)} q)$ bit operations, hence $O(\log^{2+o(1)} q)$ bit operations

and storage space altogether. Moreover, we can parametrize the set of $\mathcal{I}_{i,\vec{u}}$-good tuples as follows:

$$\mathrm{Good}_{\vec{r}_i(\vec{u})}(\mathcal{I}_{i,\vec{u}}) = \left\{(k'_p \mathfrak{d}_{i,p,\vec{u}})_{p\in\mathfrak{P}_i} : \vec{k'} = (k'_p)_{p\in\mathfrak{P}_i} \in \prod_{p\in\mathfrak{P}_i} \mathbb{Z}/\frac{l_{i,p,\vec{u}_p}}{\mathfrak{d}_{i,p,\vec{u}}}\mathbb{Z} =: K'_{i,\vec{u}}\right\}.$$

Computing $r'_{i,\vec{u}}(\vec{k'})$. Now, for each $\vec{k'} = (k'_p)_{p\in\mathfrak{P}_i} \in K'_{i,\vec{u}}$ and its associated $\mathcal{I}_{i,\vec{u}}$-good tuple $(k'_p \mathfrak{d}_{i,p,\vec{u}})_{p\in\mathfrak{P}_i}$, we have the cycle representative $(\overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p)))_{p\in\mathfrak{P}_i}$ of $\mathcal{A}'_i$, or rather, of the permutation $\bigotimes_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,p}$ identified with it, in $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$. Literally, $\mathcal{A}'_i$ is defined as an affine permutation of $\mathbb{Z}/s'_i\mathbb{Z} = \mathbb{Z}/\prod_{p\in\mathfrak{P}_i} p^{\kappa_p}\mathbb{Z}$. Therefore, the actual cycle representative of $\mathcal{A}'_i$ associated with $\vec{k'}$ is

$$r'_{i,\vec{u}}(\vec{k'}) := \sum_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p))\frac{s'_i}{p^{\kappa_p}}\,\mathrm{inv}_{p^{\kappa_p}}\left(\frac{s'_i}{p^{\kappa_p}}\right),$$

the unique element of $\mathbb{Z}/s'_i\mathbb{Z}$ that is congruent to $\overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p))$ modulo p^{κ_p} for each $p \in \mathfrak{P}_i$. We note that the expression $\overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p))$ can be spelled out explicitly as follows.

$$\overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p)) = \begin{cases} r_{i,p}(\vec{u}_p) + k'_p\mathfrak{d}_{i,p,\vec{u}}\overline{\beta}_i, & \text{if } \overline{\alpha}_i = 1, \\ \overline{\alpha}_i^{k'_p\mathfrak{d}_{i,p,\vec{u}}} r_{i,p}(\vec{u}_p) + \overline{\beta}_i \frac{\overline{\alpha}_i^{k'_p\mathfrak{d}_{i,p,\vec{u}}}-1}{\overline{\alpha}_i-1}, & \text{otherwise,} \end{cases}$$

where the fraction in the second case is to be understood as an integer division, but the expression as a whole represents an element of $\mathbb{Z}/p^{\kappa_p}\mathbb{Z}$ (one needs to identify the integer value with its reduction modulo p^{κ_p}). It takes $O(\log^{2+o(1)} q)$ bit operations and storage space to compute and store the parametric definition of $r'_{i,\vec{u}}$.

Computing L_i. At last, we can now provide the parametric definitions for the CRL-list $\mathcal{L}''_i$ of $\mathcal{A}'_i$ and, subsequently, for the CRL-list $\mathcal{L}'_i$ of $\mathcal{A}_i$. Namely, $Y_i := \bigcup_{\vec{u}\in\overline{Y}_i} (\{\vec{u}\} \times K'_{i,\vec{u}})$, and for $(\vec{u}, \vec{k'}) \in Y_i$, we set

$$\mathrm{par}''_i(\vec{u}, \vec{k'}) := (r'_{i,\vec{u}}(\vec{k'}), l_{i,\vec{u}}).$$

Then $\mathcal{L}''_i = \{\mathrm{par}''_i(\vec{u}, \vec{k'}) : (\vec{u}, \vec{k'}) \in Y_i\}$. In order to obtain par'_i and $\mathcal{L}'_i$, we simply need to lift the first entries of elements of $\mathcal{L}''_i$ (images of par''_i) from $\mathbb{Z}/s'_i\mathbb{Z}$ to $\mathbb{Z}/s\mathbb{Z}$ such that the reduction modulo s/s'_i of each lift is the unique periodic point of $\mathcal{A}_i$ mod (s/s'_i) in $\mathbb{Z}/(s/s'_i)\mathbb{Z}$. By Lemma 2.1.14, we can compute that periodic point as follows. Let L_i denote the smallest non-negative integer such that

$$\gcd(\alpha_i^{L_i}, s) = \prod_{p|\gcd(\alpha_i,s)} p^{\kappa_p},$$

which satisfies

$$L_i = \max_{p|\gcd(\overline{\alpha}_i,s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i)} \right\rceil = \max_{p|\gcd(\overline{\alpha}_i,s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i \bmod p^{\nu_p(s)})} \right\rceil \leq \mathrm{mpe}(s) \in O(\log q)$$

and may be found by computing, for each $p \mid \gcd(\overline{\alpha}_i, s)$, the value $\overline{\alpha}_i \bmod p^{\nu_p(s)}$ (for $\nu_p(s)$, one consults the factorization of s computed above), then finding $\nu_p(\overline{\alpha}_i \bmod p^{\nu_p(s)})$ with a binary search between 0 and $\nu_p(s)$ (each step of which involves a power and a gcd computation). Altogether, this costs

$$O\Big(\log q \cdot \Big(\log^{1+o(1)} q + \sum_{p|\gcd(\overline{\alpha}_i,s)} (\log\log q \cdot \log^{2+o(1)} p^{\nu_p(s)})\Big)\Big) = O(\log^{3+o(1)} q)$$

bit operations by Lemma 5.1.5 (6,8). The unique periodic point of $\mathcal{A}_i \bmod (s/s'_i)$ is the reduction of

$$\sum_{z=0}^{L_i-1} \overline{\alpha}_i^z \overline{\beta}_i = \begin{cases} L_i \overline{\beta}_i, & \text{if } \overline{\alpha}_i = 1, \\ \frac{\overline{\alpha}_i^{L_i}-1}{\overline{\alpha}_i - 1} \overline{\beta}_i, & \text{otherwise,} \end{cases}$$

modulo s/s'_i and may be computed in $O(\log^{2(1+o(1))} q) = O(\log^{2+o(1)} q)$ bit operations using that $\overline{\alpha}_i^{L_i}$ is of bit length in $O(\log^2 q)$. We obtain the following formula for par'_i (which has the domain of definition Y_i, same as par''_i) such that $\mathcal{L}'_i = \{\mathrm{par}'_i(\vec{u}, \vec{k}') : (\vec{u}, \vec{k}') \in Y_i\}$:

$$\mathrm{par}'_i(\vec{u}, \vec{k}') = \left(r'_{i,\vec{u}}(\vec{k}') \frac{s}{s'_i} \mathrm{inv}_{s'_i}\left(\frac{s}{s'_i}\right) + \sum_{z=0}^{L_i-1} \overline{\alpha}_i^z \overline{\beta}_i s'_i \, \mathrm{inv}_{s/s'_i}(s'_i), l_{i,\vec{u}} \right).$$

Printing this parametric definition of $\mathcal{L}'_i$ takes $O(\log^{2+o(1)} q)$ bits of storage space. As mentioned before, the (bijective) parametrization $\mathrm{par}_i : Y_i \to \mathcal{L}_i$ of the CRL-list $\mathcal{L}_i$ of $f_{|U_i}$ can be obtained by stretching the second entries of the images of par'_i by the factor $\ell = \ell_i$ (the $\bar{f}$-cycle length of i). That is,

$$\mathrm{par}_i(\vec{u}, \vec{k}') = \left(r'_{i,\vec{u}}(\vec{k}') \frac{s}{s'_i} \mathrm{inv}_{s'_i}\left(\frac{s}{s'_i}\right) + \sum_{z=0}^{L_i-1} \overline{\alpha}_i^z \overline{\beta}_i s'_i \, \mathrm{inv}_{s/s'_i}(s'_i), \ell \cdot l_{i,\vec{u}} \right)$$

and $\mathcal{L}_i = \{\mathrm{par}_i(\vec{u}, \vec{k}') : (\vec{u}, \vec{k}') \in Y_i\}$.

Finally, the expression $\mathrm{par}_i(\vec{u}, \vec{k}')$, where

- $i \in \mathrm{proj}_1(\overline{\mathcal{L}})$;
- $\vec{u} \in \overline{Y}_i$ (with $\overline{Y}_d := \{\emptyset\}$); and
- $\vec{k}' \in K'_{i,\vec{u}}$ (with $K'_{d,\emptyset} := \{\emptyset\}$)

forms the desired bijective parametrization of a CRL-list $\mathfrak{L}$ of f, which can be pasted together from the results of earlier computations using $O(d \log^{2+o(1)} q)$ bit operations.

In what follows, we conclude this subsection with an overview of the steps of this algorithm. At the end of the description of each step, we specify its q-bounded query complexity (QC); in the case of a loop, this is obtained component-wise by computing the sum of the entries in the corresponding components of the query complexities of the iteration steps of the loop, if applicable replacing the resulting expression by a simpler one that generates the same O-class, and multiplying it with a O-bound on the number of iterations of the loop. It follows from this overview that the query complexity of Problem 1 is as specified in statement (1) of Theorem 5.1.9, and the formula for the Las Vegas dual complexity follows from this and Lemma 5.1.7.

1 Compute the induced function $\bar{f}$ on $\{0, 1, \ldots, d\}$ and the affine maps A_i of $\mathbb{Z}/s\mathbb{Z}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Compute a CRL-list $\bar{\mathfrak{L}}$ for $\bar{f}$, storing the cycles of $\bar{f}$ in full in the process.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

3 Compute and store the parametrization $\mathrm{par}_d : Y_d \to \mathfrak{L}_d$, where $Y_d = \{(\emptyset, \emptyset)\}$ and $\mathrm{par}_d(\emptyset, \emptyset) = (0_{\mathbb{F}_q}, 1)$.
QC: $(\log d, 0, 0, 0, 0)$.

4 Compute and factor $s = (q-1)/d$.
QC: $(\log^{1+o(1)} q, 0, 1, 0, 0)$.

5 Find a primitive root r_p modulo $p^{\nu_p(s)}$ for each odd prime $p \mid s$.
QC: $(\log q, 0, 0, 0, 1)$.

6 For each $(i, \ell) \in \bar{\mathfrak{L}} \setminus \{(d, 1)\}$, with associated $\bar{f}$-cycle $(i_0, i_1, \ldots, i_{\ell-1})$, which was already computed in step 2, do the following.
QC: $(d^2 \log^{1+o(1)} q + d \log^{2+o(1)} q + \log^{3+o(1)} q, 0, d \log q, d, 0)$.

6.1 Compute the forward cycle product $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}} : z \mapsto \bar{\alpha}_i z + \bar{\beta}_i$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2 Initialize $\mathfrak{P}_i := \emptyset$.
QC: $(\log d, 0, 0, 0, 0)$.

6.3 Compute $\mathrm{ord}_{p^{\nu_p(s)}}(\bar{\alpha}_i)$ for each prime $p \mid s$.
QC: $(\log q, 0, 0, 1, 0)$.

6.4 For each prime $p \mid s$, do the following.
QC: $(\log^{2+o(1)} q, 0, \log q, 0, 0)$.

6.4.1 Check whether $p \mid \bar{\alpha}_i$, and if not, skip to the next p.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

6.4.2 Add p to $\mathfrak{P}_i$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

6.4.3 Read off $\kappa_p = \nu_p(s)$ and p^{κ_p} from the factorization of s computed in step 4.
QC: $(\log q, 0, 0, 0, 0)$.

6.4.4 Compute $\overline{\mathcal{A}}_{i,p} = \mathcal{A}_i \bmod p^{\kappa_p}$, i.e., compute $\overline{\alpha}_i \bmod p^{\kappa_p}$ and $\overline{\beta}_i \bmod p^{\kappa_p}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

6.4.5 Compute $\kappa_{i,p} := \nu_p^{(\kappa_p)}(\overline{\beta}_i) = \nu_p^{(\kappa_p)}(\overline{\beta}_i \bmod p^{\kappa_p})$, using $O(\kappa_p)$ divisions by p and increasing a counter.
QC: $(\log^{2+o(1)} p^{\kappa_p}, 0, 0, 0, 0)$.

6.4.6 If $p > 2$ then do the following.
QC: $(\log q + \log^{1+o(1)} p^{\kappa_p}, 0, 1, 0, 0)$.

6.4.6.1 Compute factorizations of $p-1$ and of $\operatorname{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)$:

$$p - 1 = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{\nu_{p,k}} \quad \text{and} \quad \operatorname{ord}_{p^{\kappa_p}}(\overline{\alpha}_i) = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{\nu'_{i,p,k}} \cdot p^{\nu'_{i,p}}.$$

QC: $(\log q, 0, 1, 0, 0)$.

6.4.6.2 Check whether $\overline{\mathcal{A}}_{i,p}$ has a fixed point and, if so, store this information and compute a fixed point $\mathfrak{f}_{i,p}$ of it according to Proposition 2.3.6. The check can be done by testing whether $\gcd((\overline{\alpha}_i \bmod p^{\kappa_p}) - 1, p^{\kappa_p})$ divides $\overline{\beta}_i \bmod p^{\kappa_p}$.
QC: $(\log^{1+o(1)} p^{\kappa_p}, 0, 0, 0, 0)$.

6.4.7 Else do the following.
QC: $(\log^{2+o(1)} 2^{\kappa_2}, 0, 0, 0, 0)$.

6.4.7.1 Compute $\nu'_{i,2} = \nu_2(\operatorname{ord}_{2^{\kappa_2}}(\overline{\alpha}_i))$ and $\nu''_{i,2} = \nu_2(\operatorname{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i))$. To avoid making another mord query, we note that $\nu''_{i,2} = \nu'_{i,2}$ unless

$$\overline{\alpha}_i \equiv \pm 1 \pmod{2^{\kappa_2}},$$

in which case $\nu''_{i,2} = 1 - \nu'_{i,2}$.
QC: $(\log^{2+o(1)} 2^{\kappa_2}, 0, 0, 0, 0)$.

6.4.7.2 Check whether $\overline{\mathcal{A}}_{i,2}$ has a fixed point and, if so, store this information and compute a fixed point $\mathfrak{f}_{i,2}$ of it (cf. step 6.4.6.2).
QC: $(\log^{1+o(1)} 2^{\kappa_2}, 0, 0, 0, 0)$.

6.4.8 Spell out a definition of the bijective parametrization $\mathrm{par}'_{i,p} : Y_{i,p} \to \mathscr{L}'_{i,p}$ of a CRL-list $\mathscr{L}'_{i,p}$ of $\overline{\mathcal{A}}_{i,p}$ in which all specified cycle lengths are fully factored, referring to Table 5.2. This requires checking which of the cases

from Table 2.2 applies, and we stored part of the information relevant for this in steps 6.4.6.2 and 6.4.7.2. We note that a general element of $Y_{i,p}$ is denoted by $\vec{u}_p$ and is either equal to u_p or (u_p, u'_p), where u_p and u'_p are integer parameters, with u_p ranging over a fixed interval, and u'_p ranging over an interval for each fixed value of u_p (with explicit formulas for the interval bounds in terms of u_p).
QC: $(\log p^{\kappa_p}, 0, 0, 0, 0)$.

6.5 Compute $s'_i = \prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.6 Compute a parametric definition of $l_{i,\vec{\vec{u}}}$, the cycle length of $\mathcal{A}'_i$ (or rather, of the permutation $\bigotimes_{p\in\mathfrak{P}_i} \bar{\mathcal{A}}_{i,p}$ on $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$ identified with it) on the point $\vec{r}_i(\vec{\vec{u}})$ represented by $\vec{\vec{u}} = (\vec{u}_p)_{p\in\mathfrak{P}_i}$. This can be done through scanning the parametric definitions of the fully factored cycle lengths

$$l_{i,p,\vec{u}_p} = \mathrm{proj}_2(\mathrm{par}'_{i,p}(\vec{u}_p)),$$

of which $l_{i,\vec{\vec{u}}}$ is the least common multiple, and performing low-cost operations on numbers of bit length in $O(\log\log q)$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.7 Set $\mathfrak{P}'_i := \mathfrak{P}_i \cup \pi(\prod_{p\in\mathfrak{P}_i}(p-1))$, using $O(\log q)$ containment checks each involving $O(1)$ copying processes of bit strings of length $O(\log q)$, and $O(\log q)$ bit comparisons and scans of memory addresses each of length $O(\log\log q)$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.8 For $\mathfrak{p} \in \mathfrak{P}'_i$, compute a parametric definition of the function value $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$ of the $\vec{r}_i(\vec{\vec{u}})$-admissible indexing function $\mathcal{I}_{i,\vec{\vec{u}}}$. This definition consists of a case distinction with at most two cases (and constant value of $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$ in each case). For $\mathfrak{p} \notin \mathfrak{P}_i$, there is always only one case, and for $\mathfrak{p} \in \mathfrak{P}_i$, the cases depend on the containment of $u_\mathfrak{p}$ in a union of certain intervals (at most five such intervals per $\mathfrak{p}$). Moreover, whenever there are two cases, one of them corresponds to $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p}) = \mathfrak{p}$. Whenever there is only one case, set $\mathcal{I}_i(\mathfrak{p}) := \mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$ for any given $\vec{\vec{u}}$, otherwise let $\mathcal{I}_i(\mathfrak{p})$ be the unique element of $\mathcal{I}_{i,\vec{\vec{u}}}(\mathfrak{p})$ that is distinct from $\mathfrak{p}$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.9 For $p \in \mathfrak{P}_i$, compute a parametric description of the pre-image set

$$\mathfrak{P}_{i,p,\vec{\vec{u}}} := \mathcal{I}^{-1}_{i,\vec{\vec{u}}}(\{p\}),$$

using formula (5.2). In this parametric description, the inclusion of primes $\mathfrak{p} \in \mathfrak{P}'_i \setminus \mathfrak{P}_i$ in $\mathfrak{P}_{i,p,\vec{\vec{u}}}$ is independent of $\vec{\vec{u}}$, whereas primes $p' \in \mathfrak{P}_i$ each have

a condition, in terms of a disjunction of bounds on $u_{p'}$ corresponding to the intervals mentioned in step 6.8, for whether $p' \in \mathfrak{P}_{i,p,\vec{\bar{u}}}$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.10 Based on step 6.9, compute parametric descriptions of

$$\mathfrak{d}_{i,p,\vec{\bar{u}}} = \prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{\bar{u}}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,\vec{\bar{u}}})} = \prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{\bar{u}}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,p,\bar{u}_p})}$$

and $l_{i,p,\vec{\bar{u}}}/\mathfrak{d}_{i,p,\vec{\bar{u}}}$ for each $p \in \mathfrak{P}_i$. In view of step 6.9, this can be achieved using suitable Kronecker deltas in the exponents.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.11 Compute the parametric description

$$r_{i,\vec{\bar{u}}}(\vec{k'}) = \sum_{p\in\mathfrak{P}_i} \bar{\mathcal{A}}_{i,p}^{k'_p \mathfrak{d}_{i,p,\vec{\bar{u}}}}(r_{i,p}(\vec{u}_p)) \frac{s'_i}{p^{\kappa_p}} \operatorname{inv}_{p^{\kappa_p}}\left(\frac{s'_i}{p^{\kappa_p}}\right)$$

of the cycle representative $r_{i,\vec{\bar{u}}}(\vec{k'})$ of $\mathcal{A}_{i'}$ associated with the parameter tuple $(\vec{\bar{u}}, \vec{k'})$, where $\vec{\bar{u}} \in \overline{Y}_i$ and

$$\vec{k'} = (k'_p)_{p\in\mathfrak{P}_i} \in K'_{i,\vec{\bar{u}}} = \prod_{p\in\mathfrak{P}_i} \mathbb{Z}/(l_{i,p,\bar{u}_p}/\mathfrak{d}_{i,p,\vec{\bar{u}}})\mathbb{Z}.$$

In this expression, the (inexplicit) affine map iterate value $\bar{\mathcal{A}}_{i,p}^{k'_p \mathfrak{d}_{i,p,\vec{\bar{u}}}}(r_{i,p}(\vec{u}_p))$ is to be substituted with the explicit formula

$$\bar{\mathcal{A}}_{i,p}^{k'_p \mathfrak{d}_{i,p,\vec{\bar{u}}}}(r_{i,p}(\vec{u}_p)) = \begin{cases} r_{i,p}(\vec{u}_p) + k'_p \mathfrak{d}_{i,p,\vec{\bar{u}}} \bar{\beta}_i, & \text{if } \bar{\alpha}_i = 1, \\ \bar{\alpha}_i^{k'_p \mathfrak{d}_{i,p,\vec{\bar{u}}}} r_{i,p}(\vec{u}_p) + \bar{\beta}_i \frac{\bar{\alpha}_i^{k'_p \mathfrak{d}_{i,p,\vec{\bar{u}}}} - 1}{\bar{\alpha}_i - 1}, & \text{otherwise.} \end{cases}$$

QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.12 Compute

$$L_i = \max_{p \mid \gcd(\bar{\alpha}_i, s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\bar{\alpha}_i)} \right\rceil = \max_{p \mid \gcd(\bar{\alpha}_i, s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\bar{\alpha}_i \bmod p^{\nu_p(s)})} \right\rceil,$$

the smallest non-negative integer such that $\gcd(\bar{\alpha}_i^{L_i}, s) = \prod_{p \mid \gcd(\bar{\alpha}_i, s)} p^{\kappa_p}$. To do so, for each $p \mid \gcd(\bar{\alpha}_i, s)$, compute $\bar{\alpha}_i \bmod p^{\nu_p(s)}$ with a division, then find $\nu_p(\bar{\alpha}_i \bmod p^{\nu_p(s)})$ with a binary search between 0 and $\nu_p(s)$.
QC: $(\log^{3+o(1)} q, 0, 0, 0, 0)$

6.13 Compute the parametric description

$$\operatorname{par}_i(\vec{\bar{u}}, \vec{k'}) = \left(r'_{i,\vec{\bar{u}}}(\vec{k'}) \frac{s}{s'_i} \operatorname{inv}_{s'_i}\left(\frac{s}{s'_i}\right) + \sum_{z=0}^{L_i - 1} \bar{\alpha}_i^z \bar{\beta}_i s'_i \operatorname{inv}_{s/s'_i}(s'_i), \ell \cdot l_{i,\vec{\bar{u}}} \right)$$

of the element of $\mathfrak{L}_i$ (a CRL-list of $f_{|U_i}$, where $U_i = \bigcup_{t=0}^{\ell-1} C_{i_t}$) associated with $(\vec{\vec{u}}, \vec{k}')$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

7 Output the parametric description $\mathrm{par}_i(\vec{\vec{u}}, \vec{k}')$ of the element of $\mathfrak{L}$ (a CRL-list of f) associated with $(i, \vec{\vec{u}}, \vec{k}')$, where $i \in \mathrm{proj}_1(\bar{\mathfrak{L}})$, $\vec{\vec{u}} \in \bar{Y}_i$ and $\vec{k}' \in K'_{i,\vec{\vec{u}}}$ (with the convention that $\bar{Y}_d = \{\emptyset\}$ and $K'_{d,\emptyset} = \{\emptyset\}$), then halt.
QC: $(d \log^{2+o(1)} q, 0, 0, 0, 0)$.

5.2.2 Proof of statement (2)

We note that the only part of our algorithm for Problem 2 where a quantum computer is required is at the beginning, when $\bar{f}$ and the A_i need to be computed. The rest of the algorithm, which we describe henceforth, uses bit operations only.

In addition to computing $\bar{f}$ and the A_i, and as at the beginning of the proof of statement (1), we need to compute the different "layers" of indices $i \in \{0, 1, \ldots, d-1\}$ according to their containment in the iterated images of $\bar{f}$, requiring $O(d \log^2 d)$ bit operations overall.

We follow the approach from Section 3.3, proceeding in three successive steps.

Step 1: transient i. We aim to compute

- $\mathcal{Z}_i = \mathcal{P}_i$ for all $\bar{f}$-transient $i \in \{0, 1, \ldots, d-1\}$;
- a list of rooted tree descriptions $(\mathfrak{D}_0, \mathfrak{D}_1, \ldots, \mathfrak{D}_{N_1})$ that covers all isomorphism types of rooted trees of the form $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ for $\bar{f}$-transient $i \in \{0, 1, \ldots, d-1\}$ and logical sign tuples $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$ such that $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset$; and
- the corresponding logical sign tuple data $S_{n,i}$.

At any given point in the algorithm (not just in this step), the set of all $n \in \mathbb{N}_0$ for which $\mathfrak{D}_n$ is defined is an initial segment $\{0, 1, \ldots, N'\}$ of $\mathbb{N}_0$, denoted by $\mathcal{N}$ (a variable that gets updated throughout the process).

In order to carry out the computations listed above, we proceed by recursion on $h_i = \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i))$. First, we assume that $h_i = 0$. Then, in accordance with Section 3.3, we set $\mathcal{P}_i := \mathfrak{P}(\emptyset)$ for all such i, introduce the trivial rooted tree isomorphism type $\mathfrak{T}_0$ via its description $\mathfrak{D}_0 := \emptyset$, and set $S_{0,i} := \{\emptyset\}$ (with $\emptyset$ to be viewed as the empty logical sign tuple), while all $S_{n,i}$ for values $n > 0$ introduced later will be defined as the empty set. This settles the case $h_i = 0$.

Now we assume that $h_i = h > 0$, and that all transient indices j with $h_j < h$ have been taken care of. The first thing we need to do for each given i is to find the $\bar{f}$-pre-images $j_1, j_2, \ldots, j_K$ of i, which requires $O(d \log d)$ bit operations. Following that, we compute a spanning congruence sequence for $\mathcal{P}_i = \bigwedge_{t=1}^{K} \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$.

This involves simple arithmetic operations (including gcd computations) and requires

$$O\left(\sum_{t=1}^{K} m_{j_t} \log^{1+o(1)} q\right) \subseteq O(d \log^{1+o(1)} q)$$

bit operations. Subsequently, we go through the logical sign tuples $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$ in lexicographic order, check whether $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset$, and if so, compute a compact description $\mathfrak{D}$ of $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$. For checking whether the block $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ is non-empty, we note that by the argument before Proposition 3.3.2, the cardinality $|\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})|$ is equal to the distribution number $\sigma_{\mathcal{P}_i,\mathbf{0}}(\vec{\nu}^{(\mathcal{P}_i)}, (\emptyset, \ldots, \emptyset))$, where $\mathbf{0}$ is the constant zero function $\mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}$. To see how costly the computation of this distribution number is, we refer to the following lemma.

Lemma 5.2.2.1. *Let $\mathcal{P}$ be an arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$, given by an explicit spanning m-congruence sequence of length $c \in \mathbb{N}^+$. Moreover, let A be an affine function $\mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$. Then for any given logical sign tuples $\vec{\nu}^{(\mathcal{P})}$ and $\vec{\nu}^{(\mathcal{P}')}$, of length c and $c+1$, respectively, it takes $O(c2^c \log^{1+o(1)} m)$ bit operations to compute the single distribution number $\sigma_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')})$.*

Proof. According to the formula in Lemma 2.2.2, computing $\sigma_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')})$ requires us to add up the summands $(-1)^{|J|}\kappa_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')}, J)$ for all $J \subseteq J_-(\vec{\nu}^{(\mathcal{P})})$, and there are $O(2^c)$ such summands. Computing a single such summand consists of

- a simple look-up of the last component of $\vec{\nu}^{(\mathcal{P}')}$ (bit operation cost: $O(\log c)$ for scanning the corresponding memory address);
- $O(c)$ integer divisibility checks following a gcd computation and subtraction, of total bit operation cost $O(c \log^{1+o(1)} m)$ by Lemma 5.1.5 (1,3,8);
- checking whether the two subsets $J_+(\vec{\nu}^{(\mathcal{P})}) \cup J$ and $J_-(\vec{\nu}^{(\mathcal{P}')})$ of $\{1, 2, \ldots, c\}$ are disjoint, which involves look-ups of entries of $\vec{\nu}^{(\mathcal{P})}$ and $\vec{\nu}^{(\mathcal{P}')}$ and takes $O(c)$ bit operations in total if pointers are used; and
- performing $O(c)$ gcd computations, integer divisions and lcm computations, of total complexity $O(c \log^{1+o(1)} m)$.

Therefore, computing all these summands $(-1)^{|J|}\kappa_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}, J)$ will take $O(c2^c \log^{1+o(1)} m)$ bit operations, which majorizes the cost of adding these summands up and is thus also the complexity of computing $\sigma_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')})$. ∎

In particular, computing $|\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})|$ to check whether that block of $\mathcal{P}_i$ is empty costs

$$O(m_i 2^{m_i} \log^{1+o(1)} q) \subseteq O(d 2^d \log^{1+o(1)} q)$$

bit operations.

Let us now assume that $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ turned out to be non-empty. Then we wish to compute a compact description $\mathfrak{D}$ of $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$. To do so, we write $\vec{\nu}^{(\mathcal{P}_i)} = \diamond_{t=1}^{K} \vec{\nu}^{(\mathcal{P}'_{j_t})}$ with $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$. By Proposition 3.3.1, we may set

$$\mathfrak{D} := \left\{ \left(n, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{n,j_t}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \right) : n \in \mathcal{N} \right\} \setminus (\mathbb{N} \times \{0\}).$$

We note that the range of the summation index in Proposition 3.3.1 includes logical sign tuples $\vec{\nu}^{(\mathcal{P}_{j_t})}$ for which $\mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ is empty, but these may be ignored (as we do here), because all corresponding distribution numbers $\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ are 0. According to Lemma 5.2.2.1, computing a single one of the distribution numbers $\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ takes

$$O(2^{m_{j_t}} m_{j_t} \log^{1+o(1)} q),$$

bit operations. Observing that for each fixed $t \in \{1, 2, \ldots, K\}$, one has

$$\bigcup \{S_{n,j_t} : n \in \mathcal{N}\} \subseteq \{\emptyset, \neg\}^{m_{j_t}},$$

we end up with a total bit operation cost of

$$O\left(\sum_{t=1}^{K} (2^{m_t} \cdot 2^{m_{j_t}} m_{j_t} \log^{1+o(1)} q) \right) \subseteq O(d 4^d \log^{1+o(1)} q)$$

for computing $\mathfrak{D}$, which majorizes the cost of checking whether $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset$. Also, if we go through the numbers $n \in \mathcal{N}$ in increasing order when computing $\mathfrak{D}$, the array representing $\mathfrak{D}$ has its elements ordered by increasing n, as it should.

Next, we need to check whether the rooted tree $\mathfrak{T}$ described by $\mathfrak{D}$ occurs among the rooted trees $\mathfrak{T}_n$, described by $\mathfrak{D}_n$, which have already been introduced. The number of those trees is at most the total number of distinct (non-empty) blocks in all $\mathcal{P}_j$, where j is $\bar{f}$-transient, and that number is in $O(\min\{d 2^d, q\})$. Since each $\mathfrak{D}_n$ as well as $\mathfrak{D}$ is a lexicographically sorted list of length in $O(\min\{d 2^d, q\})$ every entry of which is a bit string of length in $O(\log q)$, it takes $O(d^2 4^d \log q)$ bit operations to check whether $\mathfrak{D} = \mathfrak{D}_n$ for some n. Should that be the case, we add $\vec{\nu}^{(\mathcal{P}_i)}$ to $S_{n,i}$ as a new element (at the end of the array, which leads to that array being lexicographically ordered). Otherwise, we create $\mathfrak{D}$ as a new tree description $\mathfrak{D}_{n'}$, where $n' = \max \mathcal{N} + 1$, and initialize

$$S_{n',j} := \begin{cases} \{\vec{\nu}^{(\mathcal{P}_i)}\}, & \text{if } j = i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

For a given $\bar{f}$-transient i such that $h_i = h$, this loop takes

$$O(2^{m_i}(d4^d \log^{1+o(1)} q + d^2 4^d \log q))$$

bit operations. Now, distinct indices i with $h_i = h$ have disjoint iterated pre-image sets under $\bar{f}$, whence the sum of the numbers m_i for all such i is at most d. Therefore, we get a total bit operation cost of

$$O(d8^d \log^{1+o(1)} q + d^2 8^d \log q)$$

for dealing with all i such that h_i has a given value. Dealing with all $\bar{f}$-transient i in total takes

$$O(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q)$$

bit operations.

Step 2: $i = d$. Now that the $\bar{f}$-transient indices i have been taken care of, one can compute a description $\mathfrak{D}$ of $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ following Proposition 3.3.2, which is similar to a single iteration of the loop in step 1 and takes $O(d4^d \log^{1+o(1)} q)$ bit operations. Afterward, we check whether $\mathfrak{D}$ occurs among the existing descriptions $\mathfrak{D}_n$ (introduced in step 1), which (analogously to step 1) takes $O(d^2 4^d \log q)$ bit operations. If so, we set $S_{n,d} := \emptyset$ (positive logical sign) for the corresponding unique n, and $S_{m,d} := \neg$ for all other m. If not, we introduce $\mathfrak{D}$ as a new tree description $\mathfrak{D}_{n'}$, where $n' = \max \mathcal{N} + 1$, and set $S_{n',d} := \emptyset$ and $S_{m,d} := \neg$ for all $m < n'$. The overall complexity of this step is majorized by the one of step 1.

Step 3: $\bar{f}$-periodic $i < d$. Finally, we discuss $\bar{f}$-periodic indices $i \in \{0, 1, \ldots, d-1\}$. Let us use the notation from step 3 in Section 3.3. For instance, $(i_0, i_1, \ldots, i_{\ell-1})$ is the $\bar{f}$-cycle of $i = i_0$, and we have $i_t = i_{t \bmod \ell}$ for $t \in \mathbb{Z}$ as well as $i' = i_{-1}$.

We begin by computing H_i, the maximum tree height in Γ_f above periodic vertices in cosets of the form C_{i_t} for $t \in \mathbb{Z}$, for each $\bar{f}$-periodic i. Following the argument in Section 3.3, we recall that $H_i \leq \ell \,\mathrm{mpe}(s) \leq \ell \lfloor \log_2 s \rfloor$. Moreover, if we denote for fixed $k \in \mathbb{Z}$ by $h'_{i,k}$ the smallest positive integer h' such that

$$\gcd\left(\prod_{t=0}^{h'-1} \alpha_{i_{k-h'+t}}, s\right) = \gcd\left(\prod_{t=0}^{h'-2} \alpha_{i_{k-h'+t}}, s\right), \tag{5.3}$$

then

$$H_i = \max\{h'_{i,k} : k = 0, 1, \ldots, \ell - 1\} - 1.$$

Before we enter a loop over k to find $h'_{i,k}$, we compute $\bar{\alpha}_i = \prod_{t=0}^{\ell-1} \alpha_{i_t} \bmod s$, taking $O(\ell \log^{1+o(1)} s) \subseteq O(d \log^{1+o(1)} q)$ bit operations. We then enter the loop over $k = 0, 1, \ldots, \ell - 1$. For each k, we aim to find the correct value of $h'_{i,k}$ using a

binary search in the range between 1 and $\ell\lfloor \log_2 s \rfloor + 1$. Let us assume that we fixed a tentative value h'. Then for $H' \in \{h'-1, h'-2\}$, we have

$$\prod_{t=0}^{H'} \alpha_{i_{k-h'+t}} = (\overline{\alpha}_i)^{\lfloor (H'+1)/\ell \rfloor} \cdot \prod_{t=0}^{(H'+1) \bmod \ell - 1} \alpha_{i_{k-h'+t}},$$

which can be computed modulo s for both values of H' using

$$O(\log\log s \log^{1+o(1)} s + \ell \log^{1+o(1)} s) = O(\ell \log^{1+o(1)} s) \subseteq O(d \log^{1+o(1)} q)$$

bit operations. Following that, we compute and compare the two gcds from equation (5.3), which takes $O(\log^{1+o(1)} s) \subseteq O(\log^{1+o(1)} q)$ bit operations. This binary search has $O(\log(\ell \log s)) \subseteq O(\log d + \log\log q)$ iterations per k, and there are $\ell \in O(d)$ values of k to deal with. In total, the computation of H_i takes

$$O(d \cdot (\log d + \log\log q) \cdot d \log^{1+o(1)} q) = O(d^2 \log d \log^{1+o(1)} q)$$

bit operations for each individual i, and

$$O(d^3 \log d \log^{1+o(1)} q)$$

bit operations for all $\bar{f}$-periodic $i < d$ together.

After finding all H_i, we aim to compute $\mathcal{Z}_i = (\mathcal{X}_{i,h})_{h=-1,0,\ldots,H_i}$ for each $\bar{f}$-periodic i. We do so by computing $\mathcal{X}_{i,h}$ for all $\bar{f}$-periodic i together successively for $h = -1, 0, \ldots, \mathfrak{H} := \max\{H_i : i \in \mathrm{per}(\bar{f}) \setminus \{d\}\}$ (for each fixed value of h, we skip those i such that $h > H_i$). Now, $\mathcal{X}_{i,-1} = (\theta_{i,h}(x))_{h=1,2,\ldots,H_i}$ consists of H_i congruences, and according to the definition of $\theta_{i_t,h}(x)$, these congruences can be computed recursively using simple arithmetic in each step. Per i, this takes

$$\begin{aligned} O((H_i+1)\log^{1+o(1)} q) &\subseteq O((d\,\mathrm{mpe}(s)+1)\log^{1+o(1)} q) \\ &\subseteq O(d\,\mathrm{mpe}(q-1)\log^{1+o(1)} q) \end{aligned}$$

bit operations, and for all $\bar{f}$-periodic i together, it takes $O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$ bit operations to compute $\mathcal{X}_{i,-1}$. The computation of $\mathcal{X}_{i,0} = \mathcal{R}_i$ is analogous to the one of $\mathcal{Z}_j = \mathcal{P}_j$ for $\bar{f}$-transient j (see step 1), taking $O(d \log^{1+o(1)} q)$ bit operations per i, and $O(d^2 \log^{1+o(1)} q)$ for all $\bar{f}$-periodic i together. Following that, the spanning congruence sequence for

$$\mathcal{X}_{i,h} = \lambda^h_{i_{-h}}(\mathcal{R}_{i_{-h}}) = \lambda(\lambda^{h-1}_{i_{-h}}(\mathcal{R}_{i_{-h}}), A_{i_{-1}}) = \lambda(\mathcal{X}_{i_{-1},h-1}, A_{i_{-1}})$$

is obtained recursively for $h = 1, 2, \ldots, H_i$ through processing the one for $\mathcal{X}_{i_{-1},h-1}$ using simple arithmetic, again taking complexity $O(d^2 \log^{1+o(1)} q)$ in each step for

all i together. Overall, the bit operation cost of computing $\mathcal{Z}_i$ after each H_i has been worked out is in

$$\begin{aligned} &O(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q + (d \operatorname{mpe}(s) + 1) \cdot d^2 \log^{1+o(1)} q) \\ &\quad \subseteq O(d^3 \operatorname{mpe}(q-1) \log^{1+o(1)} q) \end{aligned}$$

for all $\bar{f}$-periodic $i < d$ together.

Finally, we need to

- extend the list of rooted tree descriptions $\mathfrak{D}_n$ produced in steps 1 and 2 to its final version, which additionally contains descriptions of all rooted tree isomorphism types of the form $\operatorname{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$, where $i < d$ is $\bar{f}$-periodic, $h \in \{0, 1, \ldots, H_i\}$, $\vec{\nu}^{(\mathcal{P}_{i,h})} \in \{\emptyset, \neg\}^{n_{i-h}+n_{i-h+1}+\cdots+n_{i_0}}$, and $\mathcal{B}(\mathcal{Q}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h}) \neq \emptyset$; and
- compute the associated logical sign sets $S_{n,i,h}$.

We do so recursively in the same manner as before, i.e., computing successively for $h = 0, 1, \ldots, \mathfrak{H}$ the relevant data for *all* corresponding $\bar{f}$-periodic $i < d$ together. For $h = 0$, where $\mathcal{P}_{i,h} = \mathcal{R}_i$, this is basically identical to the corresponding computations in step 1 and has the same overall bit operation cost, in $O(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q)$.

Now we assume that $h > 0$ and that all smaller values have been taken care of. We loop (in lexicographic order) over the logical sign tuples $\vec{\nu}^{(\mathcal{P}_{i,h})}$ with $\sum_{\mathfrak{t}=0}^{h} n_{i-\mathfrak{t}}$ entries, noting that there are $O(2^{(h+1)d})$ such tuples. For each tuple $\vec{\nu}^{(\mathcal{P}_{i,h})}$, we first check whether $\mathcal{B}(\mathcal{Q}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h})$ is non-empty. Because the length of the spanning congruence sequence for $\mathcal{Q}_{i,h}$ which we use here is at most $(h+1)d + H_i$, Lemma 5.2.2.1 implies that this check takes

$$O(((h+1)d + H_i) 2^{(h+1)d+H_i} \log^{1+o(1)} q)$$

bit operations. If this block of $\mathcal{Q}_{i,h}$ is indeed non-empty, we need to compute a compact description $\mathfrak{D}$ of $\operatorname{Tree}_i(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$. Let $j_1, j_2, \ldots, j_K$ be the $\bar{f}$-transient pre-images of i under $\bar{f}$ (which take $O(d \log d)$ bit operations to determine). Writing $\vec{\nu}^{(\mathcal{P}_{i,h})} = \diamond_{\mathfrak{t}=0}^{h} \vec{o'_{\mathfrak{t}}}$ with $\vec{o'_{\mathfrak{t}}} \in \{\emptyset, \neg\}^{n_{i-\mathfrak{t}}}$, and $\vec{o'_0} = \diamond_{t=1}^{K} \vec{\nu}^{(\mathcal{P}'_{j_t})}$ with $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$, we may choose $\mathfrak{D}$ as follows according to Propositions 3.3.3 and 3.3.4:

$$\begin{aligned} \mathfrak{D} := \Bigg\{ \Bigg(m, &\sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{m,j_t}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right) \\ &+ \sum_{k=0}^{h-1} \sum_{\vec{\nu}^{(\mathcal{P}_{i',k})} \in S_{m,i',k}} \sigma_{\mathcal{Q}_{i',k}, A_{i'}}\left(\vec{\nu}^{(\mathcal{P}_{i',k})} \diamond \vec{\xi}_{i',k}, \diamond_{\mathfrak{t}=1}^{k+1} \vec{o'_{\mathfrak{t}}} \diamond \vec{\xi_{i,h}}\right) \Bigg) : m \in \mathcal{N} \Bigg\} \\ \setminus (\mathbb{N} \times \{0\}). \end{aligned}$$

Computing the first sum,

$$\sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{m,j_t}} \sigma_{\mathcal{P}_{j_t},A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right),$$

for all $m \in \mathcal{N}$ *together* is analogous to the corresponding argument in step 1 (also applying Lemma 5.2.2.1 with $c := m_{j_t}$ and $m := q$) and takes $O(d4^d \log^{1+o(1)} q)$ bit operations. As for the complexity of computing the second sum,

$$\sum_{k=0}^{h-1} \sum_{\vec{\nu}^{(\mathcal{P}_{i',k})} \in S_{m,i',k}} \sigma_{\mathcal{Q}_{i',k},A_{i'}}(\vec{\nu}^{(\mathcal{P}_{i',k})} \diamond \vec{\xi}_{i',k}, \diamond_{t=1}^{k+1} \vec{o}'_t \diamond \vec{\xi}_{i,h}),$$

we note that for fixed k and $\vec{\nu}^{(\mathcal{P}_{i',k})}$, Lemma 5.2.2.1 with $c := (k+1)d + H_i$ and $m := q$ implies that computing the single distribution number

$$\sigma_{\mathcal{Q}_{i',k},A_{i'}}(\vec{\nu}^{(\mathcal{P}_{i',k})} \diamond \vec{\xi}_{i',k}, \diamond_{t=1}^{k+1} \vec{o}'_t \diamond \vec{\xi}_{i,h})$$

takes

$$O(2^{(k+1)d+H_i}((k+1)d + H_i)\log^{1+o(1)} q)$$

bit operations, whence

$$\begin{aligned} &O(2^{(k+1)d} \cdot 2^{(k+1)d+H_i}((k+1)d + H_i)\log^{1+o(1)} q) \\ &= O(2^{2(k+1)d+H_i}((k+1)d + H_i)\log^{1+o(1)} q) \end{aligned}$$

bit operations are needed for computing all of these numbers for a fixed k *and all* m *together*. Computing the second sum in its entirety for all m together takes

$$\begin{aligned} &O\left(\sum_{k=0}^{h-1} 2^{2(k+1)d+H_i}((k+1)d + H_i)\log^{1+o(1)} q\right) \\ &\subseteq O(2^{2hd+H_i}(hd + H_i)\log^{1+o(1)} q), \end{aligned}$$

bit operations, which majorizes the overall bit operation costs for computing the first sum and for checking whether $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$, and thus is also the cost of computing $\mathfrak{D}$.

Next, we need to check if $\mathfrak{D}$ occurs among the already introduced descriptions $\mathfrak{D}_n$ (for $n \in \mathcal{N}$). The number $|\mathcal{N}|$ of these descriptions is at most the sum of the numbers of distinct (non-empty) blocks in arithmetic partitions of one of the forms

- $\mathcal{P}_j$, where j is $\bar{f}$-transient, or
- $\mathcal{P}_{j,k}$, where $j < d$ is $\bar{f}$-periodic and $k \leq h$,

and that sum is in

$$O\left(d2^d + d\sum_{k=0}^{h} 2^{(k+1)d}\right) = O(d2^{(h+1)d}).$$

Moreover, each of the descriptions $\mathfrak{D}_n$ for $n \in \mathcal{N}$ as well as $\mathfrak{D}$ is a lexicographically sorted list of length in $O(\min\{d2^{(h+1)d}, q\})$ each entry of which is a bit string of length in $O(\log q)$, so it takes $O(d^2 4^{(h+1)d} \log q)$ bit operations to check if $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$. If so, we add $\vec{v}^{(\mathcal{P}_{i,h})}$ to $S_{n,i,h}$ as a new element. Otherwise, we create $\mathfrak{D}$ as a new tree description $\mathfrak{D}_{n'}$, where $n' = \max \mathcal{N} + 1$, and initialize

$$S_{n',j,h} := \begin{cases} \{\vec{v}^{(\mathcal{P}_{i,h})}\}, & \text{if } j = i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

For a given $\bar{f}$-periodic $i < d$, this loop has a bit operation cost in

$$\begin{aligned} &O(2^{(h+1)d} \cdot (2^{2hd+H_i}(hd + H_i)\log^{1+o(1)} q + 4^{(h+1)d} d^2 \log q)) \\ &\quad \subseteq O(2^{3hd+H_i+d}(hd + H_i)\log^{1+o(1)} q + 8^{(h+1)d} d^2 \log q), \end{aligned}$$

and doing this for all such i for a fixed value of h costs

$$O(d(hd + \mathfrak{H})2^{3hd+\mathfrak{H}+d} \log^{1+o(1)} q + d^3 8^{(h+1)d} \log q)$$

bit operations. In total, the bit operation cost of step 3 is in

$$\begin{aligned} &O(d^3 \operatorname{mpe}(q-1)\log^{1+o(1)} q + d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q \\ &+ \sum_{h=1}^{\mathfrak{H}} (d(hd + \mathfrak{H})2^{3hd+\mathfrak{H}+d} \log^{1+o(1)} q + d^3 8^{(h+1)d} \log q)) \\ &\quad \subseteq O(d^3 \operatorname{mpe}(q-1)\log^{1+o(1)} q + d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q \\ &\qquad + d(d+1)\mathfrak{H} 2^{3\mathfrak{H}d+\mathfrak{H}+d} \log^{1+o(1)} q + d^3 8^{(\mathfrak{H}+1)d} \log q) \\ &\quad \subseteq O(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d)\operatorname{mpe}(q-1)+d} \log^{1+o(1)} q + d^3 8^{(d \operatorname{mpe}(q-1)+1)d} \log q) \\ &\quad \subseteq O(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d)\operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q), \end{aligned}$$

which majorizes the costs of steps 1 and 2 and thus is the overall bit operation cost of the algorithm for Problem 2, as asserted. The claims on the length $|\mathcal{N}|$ of the constructed recursive tree description list, as well as on the memory costs of the individual rooted tree descriptions, can be deduced as follows. Through applying the above bound on $|\mathcal{N}|$ that holds throughout the h-th iteration of the loop with $h = \mathfrak{H} \leq d \operatorname{mpe}(q-1)$, we get

$$|\mathcal{N}| \in O(d2^{(\mathfrak{H}+1)d}) \subseteq O(d2^{(d \operatorname{mpe}(q-1)+1)d}) = O(d2^{d^2 \operatorname{mpe}(q-1)+d}).$$

Moreover, $|\mathcal{N}| \leq q$ because by construction, each rooted tree isomorphism type in the associated recursive tree description list is of the form $\mathrm{Tree}_{\Gamma_f}(x)$ for some $x \in \mathbb{F}_q = \mathrm{V}(\Gamma_f)$. Each individual tree description $\mathfrak{D}_n$ is a set consisting of pairs of the form (m, k), where the first entries m are pairwise distinct elements of $\mathcal{N}$, whence the length of $\mathfrak{D}_n$ as a list is at most $|\mathcal{N}| \in O(\min\{d2^{d^2\,\mathrm{mpe}(q-1)+d}, q\})$. Finally, the bit cost of storing an individual pair (m, k) is in $O(\log q)$, as required.

We conclude this subsection with a detailed overview of the steps of this algorithm in the form of pseudocode, using the same format as at the end of Section 5.2.1.

1 Compute the induced function $\bar{f}$ on $\{0, 1, \ldots, d\}$ and the affine maps A_i of $\mathbb{Z}/s\mathbb{Z}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 For $i \in \{0, 1, \ldots, d-1\}$, let

$$h_i := \begin{cases} \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i)), & \text{if } i \text{ is } \bar{f}\text{-transient}, \\ \infty, & \text{otherwise.} \end{cases}$$

For each attainable value h of h_i, compute the associated list Layer_h of indices i.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

3 Set $\mathcal{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

4 Set $\vec{\mathfrak{D}} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

5 Set $\vec{\imath} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

6 For $h = 0, 1, \ldots, \max\{h_i : i \in \{0, 1, \ldots, d-1\} \setminus \mathrm{per}(\bar{f})\}$, do the following.
QC: $(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q, 0, 0, 0, 0)$.

6.1 If $h = 0$, then do the following.

6.1.1 Set $N' := 0$.
QC: $(1, 0, 0, 0, 0)$.

6.1.2 Add N' to $\mathcal{N}$ as a new element.
QC: $(1, 0, 0, 0, 0)$.

6.1.3 Set $\mathfrak{D}_0 := \emptyset$, and add it to $\vec{\mathfrak{D}}$ as a new element.
QC: $(1, 0, 0, 0, 0)$.

6.1.4 For each $i \in \mathrm{Layer}_0$, do the following.
QC: $(d \log d, 0, 0, 0, 0)$.

6.1.4.1 Set $\mathcal{Z}_i := \mathcal{P}_i := \mathfrak{P}(\emptyset)$ and $m_i := 0$.
QC: $(\log d, 0, 0, 0, 0)$.

6.1.4.2 Set $S_{0,i} := \{\emptyset\}$.
QC: $(\log d, 0, 0, 0, 0)$.

6.1.4.3 Add i to $\vec{\imath}$ as a new element.
QC: $(\log d, 0, 0, 0, 0)$.

6.2 Else do the following.

6.2.1 For each $i \in \mathrm{Layer}_h$, do the following.
QC: $(d8^d \log^{1+o(1)} q + d^2 8^d \log q, 0, 0, 0, 0)$.

6.2.1.1 Compute the $\bar{f}$-pre-images $j_1, j_2, \ldots, j_K$ of i.
QC: $(d \log d, 0, 0, 0, 0)$.

6.2.1.2 Compute a spanning congruence sequence, of length m_i, for $\mathcal{Z}_i := \mathcal{P}_i := \bigwedge_{t=1}^{K} \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2.1.3 For each $n \in \mathcal{N}$, do the following.
QC: $(d2^d \log q, 0, 0, 0, 0)$.

6.2.1.3.1 Initialize $S_{n,i} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

6.2.1.4 For each $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$, do the following.
QC: $(2^{m_i}(d4^d \log^{1+o(1)} q + d^2 4^d \log q), 0, 0, 0, 0)$.

6.2.1.4.1 Check whether $|\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})| = \sigma_{\mathcal{P}_i,\emptyset}(\vec{\nu}^{(\mathcal{P}_i)}, (\emptyset, \ldots, \emptyset)) = 0$, and if so, skip to the next tuple $\vec{\nu}^{(\mathcal{P}_i)}$.
QC: $(d2^d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2.1.4.2 Writing $\vec{\nu}^{(\mathcal{P}_i)} = \diamond_{t=1}^{K} \vec{\nu}^{(\mathcal{P}'_{j_t})}$ with $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$, compute

$$\mathfrak{D} := \left\{ \left(n, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{n,j_t}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \right) : n \in \mathcal{N} \right\} \setminus (\mathbb{N} \times \{0\}),$$

a compact description of $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$.
QC: $(d4^d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2.1.4.3 Check whether $\mathfrak{D} = \mathfrak{D}_n$ for some (unique) $n \in \mathcal{N}$, and store this information (the truth value and, if applicable, n).
QC: $(d^2 4^d \log q, 0, 0, 0, 0)$.

6.2.1.4.4 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

6.2.1.4.4.1 Add $\vec{\nu}^{(\mathcal{P}_i)}$ to $S_{n,i}$ as a new element.
QC: $(\log q + d, 0, 0, 0, 0)$.

6.2.1.4.5 Else do the following.

6.2.1.4.5.1 Set $N' := N' + 1$, and add it to $\mathcal{N}$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

6.2.1.4.5.2 Set $\mathfrak{D}_{N'} := \mathfrak{D}$, and add it to $\vec{\mathfrak{D}}$ as a new element.
QC: $(d\,2^d \log q, 0, 0, 0, 0)$.

6.2.1.4.5.3 For $j \in \vec{\imath}$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

6.2.1.4.5.3.1 Set $S_{N',j} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

6.2.1.4.5.4 Set $S_{N',i} := \{\vec{v}^{(\mathcal{P}_i)}\}$.
QC: $(\log q + d, 0, 0, 0, 0)$.

6.2.1.5 Add i to $\vec{\imath}$ as a new element.
QC: $(\log d, 0, 0, 0, 0)$.

7 Compute the $\bar{f}$-transient $\bar{f}$-pre-images $j_1, j_2, \ldots, j_K$ of d.
QC: $(d \log d, 0, 0, 0, 0)$.

8 Compute

$$\mathfrak{D} := \left\{ \left(n, \sum_{t=1}^{K} \sum_{\vec{v}^{(\mathcal{P}_{j_t})} \in S_{n,j_t}} \sigma_{\mathcal{P}_{j_t},\mathbf{0}}(\vec{v}^{(\mathcal{P}_{j_t})}, (\emptyset, \ldots, \emptyset)) \right) : n \in \mathcal{N} \right\} \setminus (\mathbb{N} \times \{0\}),$$

a compact description of $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$.
QC: $(d\,4^d \log^{1+o(1)} q, 0, 0, 0, 0)$.

9 Check whether $\mathfrak{D} = \mathfrak{D}_n$ for some (unique) $n \in \mathcal{N}$, and store this information (the truth value and, if applicable, n).
QC: $(d^2 4^d \log q, 0, 0, 0, 0)$.

10 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

10.1 Set $S_{n,d} := \emptyset$, and $S_{m,d} := \neg$ for all $m \in \mathcal{N} \setminus \{n\}$.
QC: $(d\,2^d \log q, 0, 0, 0, 0)$.

11 Else do the following.

11.1 Set $N' := N' + 1$, and add it to $\mathcal{N}$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

11.2 Set $\mathfrak{D}_{N'} := \mathfrak{D}$, and add it to $\vec{\mathfrak{D}}$ as a new element.
QC: $(d\,2^d \log q, 0, 0, 0, 0)$.

11.3 For $j \in \vec{\imath} = \{0, 1, \ldots, d-1\} \setminus \mathrm{per}(\bar{f})$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

11.3.1 Set $S_{N',j} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

11.4 Set $S_{N',d} := \emptyset$, and $S_{n,d} := \neg$ for all $n \in \mathcal{N} \setminus \{N'\}$.
QC: $(d\,2^d \log q, 0, 0, 0, 0)$.

12 For $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d^3 \log d \log^{1+o(1)} q, 0, 0, 0, 0)$.

12.1 Compute H_i, the maximum tree height in Γ_f above periodic vertices in cosets of the form C_{i_t}, where $t \in \mathbb{Z}$. See the discussion above for details.
QC: $(d^2 \log d \log^{1+o(1)} q, 0, 0, 0, 0)$.

13 Compute $\mathfrak{H} := \max\{H_i : i \in \mathrm{per}(\bar{f}) \setminus \{d\}\}$.
QC: $(d \log d, 0, 0, 0, 0)$.

14 For $h = -1, 0, 1, \ldots, \mathfrak{H}$, do the following.
QC: $(d^3\,\mathrm{mpe}(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.1 If $h = -1$, then do the following.

14.1.1 For $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d^2\,\mathrm{mpe}(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.1.1.1 Compute and store the $\bar{f}$-pre-images of i, including the periodic one, i'.
QC: $(d \log d, 0, 0, 0, 0)$.

14.1.1.2 Compute $\mathcal{X}_{i,-1} := (\theta_{i,k}(x))_{k=1,2,\ldots,H_i}$.
QC: $(d\,\mathrm{mpe}(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2 Else do the following.

14.2.1 If $h = 0$, then do the following.

14.2.1.1 For $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d^2 \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2.1.1.1 Compute $\mathcal{X}_{i,0} = \mathcal{R}_i = \bigwedge_{t=1}^{K} \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$, with a spanning sequence of length n_i, where $j_1, j_2, \ldots, j_K$ are the $\bar{f}$-transient $\bar{f}$-pre-images of i (this can be handled analogously to steps 6.2.1.1 and 6.2.1.2).
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2.2 Else do the following.

14.2.2.1 For $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d^2 \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2.2.1.1 If $h \le H_i$, then compute $\mathcal{X}_{i,h} = \lambda(\mathcal{X}_{i',h-1}, A_{i'})$ (we observe that as a spanning congruence sequence, $\mathcal{X}_{i,h}$ has length $n_{i'}$, same as $\mathcal{X}_{i',h-1}$). Otherwise, skip to the next value of i.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

15 For $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d^3\,\mathrm{mpe}(q-1) \log q, 0, 0, 0, 0)$.

15.1 Set $\mathcal{Z}_i := (\mathcal{X}_{i,h})_{h=-1,0,\ldots,H_i}$.
QC: $(d^2 \operatorname{mpe}(q-1)\log q, 0, 0, 0, 0)$.

16 For $h = 0, 1, \ldots, \mathfrak{H}$, do the following.
QC: $(d^3 \operatorname{mpe}(q-1)2^{(3d^2+d)\operatorname{mpe}(q-1)+2d}\log^{1+o(1)} q, 0, 0, 0, 0)$.

16.1 Initialize all sets $S_{n,i,h}$, where $n \in \mathcal{N}$ and $i \in \operatorname{per}(\bar{f}) \setminus \{d\}$ such that $H_i \geq h$, to be $\emptyset$.
QC: $(d^2 2^{hd}, 0, 0, 0, 0)$.

16.2 If $h = 0$, then do the following.

16.2.1 Extend $\mathcal{N}$ and the associated list $\vec{\mathfrak{D}}$ of rooted tree descriptions $\mathfrak{D}_n$ such that all rooted tree isomorphism types of the form

$$\operatorname{Tree}_i(\mathcal{P}_{i,0}, \vec{\nu}^{(\mathcal{P}_{i,0})}) = \operatorname{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{P}_{i,0})}\right)$$

for $i \in \operatorname{per}(\bar{f}) \setminus \{d\}$ are covered, and compute the associated logical sign tuple sets $S_{n,i,0}$. This is analogous to step 6.2.1.4, but carried out for the $O(d)$ values of $i \in \operatorname{per}(\bar{f}) \setminus \{d\}$ together.
QC: $(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q, 0, 0, 0, 0)$.

16.3 Else do the following.

16.3.1 For each $i \in \operatorname{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d(hd + \mathfrak{H})2^{3hd+\mathfrak{H}+d}\log^{1+o(1)} q + d^3 8^{(h+1)d}\log q, 0, 0, 0, 0)$.

16.3.1.1 Check whether $h \leq H_i$. If not, skip to the next i.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2 Recalling that $\mathcal{P}_{i,h} = \bigwedge_{t=0}^{h} \mathcal{X}_{i,t}$, do the following for each $\vec{\nu}^{(\mathcal{P}_{i,h})} \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}}$.
QC: $((hd + H_i)2^{3hd+H_i+d}\log^{1+o(1)} q + d^2 8^{(h+1)d}\log q, 0, 0, 0, 0)$.

16.3.1.2.1 Recalling that $\mathcal{Q}_{i,h} = \mathcal{P}_{i,h} \wedge \mathcal{U}_i$, where $\mathcal{U}_i = \mathfrak{P}'(\theta_{i,k}(x) : k = 1, 2, \ldots, H_i)$ (and the definition of $\vec{\xi}_{i,h} \in \{\emptyset, \neg\}^{H_i}$ from page 55), check whether

$$|\mathcal{B}(\mathcal{Q}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h})| = \sigma_{\mathcal{Q}_{i,h},\mathbf{0}}(\vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h}, (\emptyset, \ldots, \emptyset)) = 0,$$

and if so, skip to the next tuple $\vec{\nu}^{(\mathcal{P}_{i,h})}$.
QC: $(((h+1)d + H_i)2^{(h+1)d+H_i}\log^{1+o(1)} q, 0, 0, 0, 0)$.

16.3.1.2.2 Writing $\vec{\nu}^{(\mathcal{P}_{i,h})} = \diamond_{t=0}^{h} \vec{o'_t}$ with $\vec{o'_t} \in \{\emptyset, \neg\}^{n_{i_{-t}}}$, and

$$\vec{o'_0} = \diamond_{t=1}^{K} \vec{\nu}^{(\mathcal{P}'_{j_t})}$$

with $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{n_{j_t}+1}$ (where $j_1, j_2, \ldots, j_K$ are the $\bar{f}$-transient $\bar{f}$-pre-images of i, computed in step 14.1.1.1), compute

$$\mathfrak{D} := \left\{ \left(m, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{m,j_t}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) + \sum_{k=0}^{h-1} \sum_{\vec{\nu}^{(\mathcal{P}_{i',k})} \in S_{m,i',k}} \sigma_{\mathcal{Q}_{i',k}, A_{i'}}(\vec{\nu}^{(\mathcal{P}_{i',k})} \diamond \vec{\xi}_{i',k}, \diamond_{\mathfrak{t}=1}^{k+1} \vec{o}'_{\mathfrak{t}} \diamond \vec{\xi}_{i,h}) \right) : m \in \mathcal{N} \right\} \setminus (\mathbb{N} \times \{0\}).$$

QC: $(2^{2hd+H_i}(hd + H_i)\log^{1+o(1)} q, 0, 0, 0, 0)$.

16.3.1.2.3 Check whether $\mathfrak{D} = \mathfrak{D}_n$ for some (unique) $n \in \mathcal{N}$, and store this information (the truth value and, if applicable, n).
QC: $(d^2 4^{(h+1)d} \log q, 0, 0, 0, 0)$.

16.3.1.2.4 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

16.3.1.2.4.1 Add $\vec{\nu}^{(\mathcal{P}_{i,h})}$ to $S_{n,i,h}$ as a new element.
QC: $(\log q + (h+1)d, 0, 0, 0, 0)$.

16.3.1.2.5 Else do the following.

16.3.1.2.5.1 Set $N' := N' + 1$, and add it to $\mathcal{N}$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2.5.2 Set $\mathfrak{D}_{N'} := \mathfrak{D}$, and add it to $\vec{\mathfrak{D}}$ as a new element.
QC: $(d 2^{(h+1)d} \log q, 0, 0, 0, 0)$.

16.3.1.2.5.3 For $j \in \vec{\imath} = \{0, 1, \ldots, d-1\} \setminus \mathrm{per}(\bar{f})$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

16.3.1.2.5.3.1 Set $S_{N',j} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2.5.4 Set $S_{N',d} := \neg$.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2.5.5 For $k = 0, 1, \ldots, h-1$, do the following.
QC: $(dh \log q, 0, 0, 0, 0)$.

16.3.1.2.5.5.1 For $j \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

16.3.1.2.5.5.1.1 Set $S_{N',j,k} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2.5.6 For $j \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d \log q + (h+1)d^2, 0, 0, 0, 0)$.

16.3.1.2.5.6.1 Set

$$S_{N',j,h} := \begin{cases} \{\vec{v}^{(\mathcal{P}_{i,h})}\}, & \text{if } j = i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

QC: $(\log q + (h+1)d, 0, 0, 0, 0)$.

17 For $n \in \mathcal{N}$, do the following.
QC: $(d^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+d}, 0, 0, 0, 0)$; please note that although the QC term for step 16 only involves d^3, not d^4, it still majorizes this.

17.1 For $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$, do the following.
QC: $(d^3 \operatorname{mpe}(q-1)2^{d^2 \operatorname{mpe}(q-1)+d}, 0, 0, 0, 0)$.

17.1.1 Set $S_{n,i} := (S_{n,i,h})_{h=0,1,\ldots,H_i}$.
QC: $(d^2 \operatorname{mpe}(q-1)2^{d^2 \operatorname{mpe}(q-1)+d}, 0, 0, 0, 0)$ for copying, using that

$$\begin{aligned} \sum_{h=0}^{H_i} 2^{(h+1)d}(h+1)d &\in O((H_i+1)d2^{(H_i+1)d}) \\ &\subseteq O(d^2 \operatorname{mpe}(q-1)2^{d^2 \operatorname{mpe}(q-1)+d}) \end{aligned}$$

18 Output the partition-tree register $((\mathcal{Z}_i)_{i=0,1,\ldots,d-1}, ((\mathfrak{D}_n, (S_{n,i})_{i=0,1,\ldots,d}))_{n\in\mathcal{N}})$, and halt.
QC: $(d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log q + d^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+d}, 0, 0, 0, 0)$ for copying, using that the bit storage cost of $\mathcal{X}_{i,-1} = (\theta_{i,h}(x))_{h=1,2,\ldots,H_i}$ for fixed i is in

$$O(H_i \log q) \subseteq O(d \operatorname{mpe}(q-1) \log q),$$

the bit storage cost of $\mathcal{X}_{i,h}$ for fixed i and h is in $O((h+1)d \log q)$, the total bit storage cost of the recursive tree description list $(\mathfrak{D}_n)_{n\in\mathcal{N}}$ is in $O(|\mathcal{N}|^2 \log q) \subseteq O(d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log q)$, the total bit storage cost of the $S_{n,i}$ for $n \in \mathcal{N}$ and $i \in \{0, 1, \ldots, d\}$ is in $O(d^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+d})$ (the bit operation cost of step 17; the combined storage cost of the $S_{n,i}$ for $i \notin \mathrm{per}(\bar{f})$ or $i = d$ is majorized by that for $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$), and

$$\begin{aligned} &O(dH_i \log q + d \sum_{h=0}^{H_i} (h+1)d \log q + d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log q \\ &\quad + d^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+d}) \\ &\quad \subseteq O(d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log q + d^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+d}). \end{aligned}$$

5.2.3 Proof of statement (3)

We follow the approach of Section 3.4. If $r = 0_{\mathbb{F}_q}$, then we search for the unique $n \in \mathcal{N} = \{0, 1, \ldots, N\}$ such that $S_{n,d} = \emptyset$ (as opposed to $\neg$). This process takes $O(N) \subseteq O(d\,2^{d^2\,\mathrm{mpe}(q-1)+d})$ bit operations – see statement (2) for this bound on N. Once n has been found, one may simply read off the description $\mathfrak{D}_n$ of $\mathfrak{T}_n = \mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ and output the description $[\mathfrak{D}_n]$ of the cyclic sequence in question, which takes $O(d\,2^{d^2\,\mathrm{mpe}(q-1)+d}\log q)$ bit operations for copying.

Henceforth, we assume that $r \neq 0_{\mathbb{F}_q}$. We recall from the proof of statement (4) of Lemma 5.1.6 that there is a natural choice ω for a primitive element of $\mathbb{F}_q$. We compute $\mathfrak{l} := \log_\omega(r)$ in a single fdl query (if (r, l) is obtained as an element of the CRL-list of f that is parametrized by the algorithm for Problem 1, then r is literally specified as a power of ω, and this computation may be skipped). Then we set $i := \mathfrak{l} \bmod d$ (taking $O(\log^{1+o(1)} q)$ bit operations to compute by Lemma 5.1.5 (3)), so that $r \in C_i$, and determine the cycle $(i_0, i_1, \ldots, i_{\ell-1})$ of $i = i_0$ under $\bar{f}$, which costs $O(d \log d)$ bit operations (see also the beginning of the argument for statement (1)). Moreover, we set

$$r'_{i_0} = r'_i := \iota_i^{-1}(r) = \frac{\mathfrak{l} - i}{d} \in \mathbb{Z}/s\mathbb{Z} \quad \text{and} \quad r'_{i_t} := A_{i_{t-1}}(r'_{i_{t-1}}) \text{ for } t = 1, 2, \ldots, \ell - 1,$$

which takes $O(d \log^{1+o(1)} q)$ bit operations altogether. Finally, for $t = 0, 1, \ldots, \ell - 1$, we compute

$$\mathcal{A}_{i_t} := A_{i_t} A_{i_{(t+1) \bmod \ell}} \cdots A_{i_{(t+\ell-1) \bmod \ell}},$$

which takes $O(d^2 \log^{1+o(1)} q)$ bit operations for all t together (see also the beginning of the argument for statement (1)).

Until further notice, we assume that t is fixed. In the notation of Section 3.4, we have $\mathcal{A}_{i_t}(x) = \overline{\alpha}_{i_t} x + \overline{\beta}_{i_t}$. We compute $\gcd(\overline{\alpha}_{i_t}, s)$, find the smallest non-negative integer L_{i_t} such that $\gcd(\overline{\alpha}_{i_t}^{L_{i_t}}, s) = \gcd(\overline{\alpha}_{i_t}^{L_{i_t}+1}, s) =: s''_{i_t}$, compute $s'_{i_t} := s/s''_{i_t}$ and $\mathfrak{f}_{i_t} := \sum_{z=0}^{L_{i_t}-1} \overline{\alpha}_{i_t}^z \overline{\beta}_{i_t} \bmod s''_{i_t}$ (the unique periodic point of the reduction of $\mathcal{A}_{i_t}$ modulo s''_{i_t}). Together, these tasks can be performed within $O(\log^{3+o(1)} q)$ bit operations if binary search is used to find L_{i_t} (see also the paragraph in the proof of statement (1), where L_i is introduced).

In what follows, we focus on the $\mathcal{A}_{i_t}$-cycle of r'_{i_t} (which corresponds to the cycle of $f^t(r)$ under f^ℓ). For each $j \in \{1, 2, \ldots, m_{i_t}\}$, we compute the affine discrete logarithm

$$\mathfrak{l}_{i_t,j} := \log_{\mathcal{A}_{i_t}}^{(\mathfrak{a}_{i_t,j})}(r'_{i_t}, \mathfrak{b}_{i_t,j})$$

(as defined in Section 2.4), which takes $O(m_{i_t}) \subseteq O(d^2\,\mathrm{mpe}(q-1))$ mdl queries and $O(m_{i_t} \log^{1+o(1)} q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$ bit operations by the reduction

argument in Section 2.4. For the bound $m_{i_t} \in O(d^2 \operatorname{mpe}(q-1))$, we recall from Section 3.3 that

$$\mathcal{P}_{i_t} = \mathcal{Q}_{i_t, H_{i_t}} = \bigwedge_{k=0}^{H_{i_t}} \lambda_{i_{t-k}}^k(\mathcal{R}_{i_{t-k}}) \wedge \mathcal{U}_{i_t},$$

and that the length of the "standard" spanning congruence sequence of $\lambda_{i_{t-k}}^k(\mathcal{R}_{i_{t-k}})$ (stored in the partition-tree register as $\mathcal{X}_{i_t,k}$) is $n_{i_{t-k}} \in O(d)$, while the one of $\mathcal{U}_{i_t} = \mathfrak{P}(\theta_{i_t,h}(x) : h = 0, 1, \ldots, H_{i_t})$, which is stored as $\mathcal{X}_{i_t,-1}$, has length H_i. Moreover, $H_i \in O(d \operatorname{mpe}(q-1))$.

In addition to computing $\mathfrak{l}_{i_t,j}$ for every $j \in \{1, 2, \ldots, m_{i_t}\}$, we also compute, for each j such that $\mathfrak{l}_{i_t,j} < \infty$, the cycle length $l_{i_t,j}$ of r'_{i_t} under $\mathcal{A}_{i_t}$ modulo $\mathfrak{a}_{i_t,j}$ – again, by the reduction argument of Section 2.4, this takes $O(m_{i_t}) \subseteq O(d^2 \operatorname{mpe}(q-1))$ mord queries and $O(m_{i_t} \log^{1+o(1)} q) \subseteq O(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q)$ bit operations.

Now, observe that by the definition of the affine discrete logarithm, for each index $j \in \{1, 2, \ldots, m_{i_t}\}$, the equality $\mathfrak{l}_{i_t,j} = \infty$ is equivalent to the j-th spanning congruence of $\mathcal{P}_{i_t}$, $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$, being constantly false along the $\mathcal{A}_{i_t}$-cycle of r'_{i_t}. In that case, we say that the index j is *of type* I and set $v_j := \neg$.

Moreover, if $\mathfrak{l}_{i_t} < \infty$, then the congruence $x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$ is true for exactly one point on the *modular* $\mathcal{A}_{i_t}$-cycle of r'_{i_t} modulo $\mathfrak{a}_{i_t,j}$. Hence, for each index $j \in \{1, 2, \ldots, m_{i_t}\}$, this congruence is constantly true along the (non-modular) $\mathcal{A}_{i_t}$-cycle of r'_{i_t} if and only if $\mathfrak{l}_{i_t,j} < \infty$ and $l_{i_t,j} = 1$. In that case, we say that the index j is *of type* II and set $v_j := \emptyset$.

An index $j \in \{1, 2, \ldots, m_{i_t}\}$ that is neither of type I nor of type II is said to be *of type* III. For such indices, we leave the value of v_j open. Note that in the notation of Section 3.4, the type-III indices $j \in \{1, 2, \ldots, m_{i_t}\}$ are precisely the elements of the set I_{i_t} (of indices for which the truth value of the j-th spanning congruence of $\mathcal{P}_{i_t}$ is constant along the $\mathcal{A}_{i_t}$-cycle of r'_{i_t}).

We note that there are other, sufficient but not necessary conditions for the j-th spanning congruence of $\mathcal{P}_{i_t}$ having a constant truth value along the $\mathcal{A}_{i_t}$-cycle of r'_{i_t}, based on the observation (already made in Section 3.4) that each point on that cycle is congruent to $\mathfrak{f}_{i_t}$ modulo s''_{i_t}:

- if $\mathfrak{f}_{i_t} \not\equiv \mathfrak{b}_{i_t,j} \pmod{\gcd(\mathfrak{a}_{i_t,j}, s''_{i_t})}$, then the j-th spanning congruence of $\mathcal{P}_{i_t}$ is always false along the cycle.
- if $\mathfrak{a}_{i_t,j} \mid s''_{i_t}$ and $\mathfrak{f}_{i_t} \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}$, then the j-th spanning congruence of $\mathcal{P}_{i_t}$ is always true along the cycle.

Those conditions do not require a quantum computer to be checked efficiently, and in practice, they may speed up an implementation of the algorithm we are currently discussing. However, they do not improve the worst-case complexity of the algorithm, and so we will ignore them in this theoretical discussion.

We recall that the f-cycle length l of r is given to us as part of the input. Associated with each $j \in \{1, 2, \ldots, m_{i_t}\} \setminus I_{i_t}$, we have the (l/ℓ)-congruence $y \equiv \mathfrak{l}_{i_t,j} \pmod{l_{i_t,j}}$, which holds precisely for those $y \in \mathbb{Z}/(l/\ell)\mathbb{Z}$ such that the j-th spanning congruence of $\mathcal{P}_{i_t}$ becomes true when substituting $x := \mathcal{A}_{i_t}^{y}(r'_{i_t})$. As in Section 3.4, we set

$$\mathcal{P}^{(i_t)} := \mathfrak{P}(y \equiv \mathfrak{l}_{i_t,j} \pmod{l_{i_t,j}} : j \in \{1, 2, \ldots, m_{i_t}\} \setminus I_{i_t}),$$

an arithmetic partition of $\mathbb{Z}/(l/\ell)\mathbb{Z}$. The logical sign tuples which parametrize the blocks of $\mathcal{P}^{(i_t)}$ are from the set $\{\emptyset, \neg\}^{\{1,2,\ldots,m_{i_t}\}\setminus I_{i_t}}$ and are of the form

$$\vec{\nu} = (\nu_j)_{j \in \{1,\ldots,m_{i_t}\}\setminus I_{i_t}}.$$

For such a tuple $\vec{\nu}$, we set $\vec{\nu}^{+} := (\nu_j)_{j=1,2,\ldots,m_{i_t}}$ (the positions indexed by $j \in I_{i_t}$ are filled with the constant logical signs for such j that were defined above). Conversely, for $\vec{\nu'} \in \{\emptyset, \neg\}^{m_{i_t}}$, we denote by $\vec{\nu'}^{-}$ the projection of $\vec{\nu'}$ to $\{\emptyset, \neg\}^{\{1,2,\ldots,m_{i_t}\}\setminus I_{i_t}}$.

It follows that if $y \in \mathbb{Z}/(l/\ell)\mathbb{Z}$ is contained in the block $\mathcal{B}(\mathcal{P}^{(i_t)}, \vec{\nu})$ of $\mathcal{P}^{(i_t)}$, then $\mathcal{A}_{i_t}^{y}(r'_{i_t})$ is always contained in the block $\mathcal{B}(\mathcal{P}_{i_t}, \vec{\nu}^{+})$ of $\mathcal{P}_{i_t}$, whence

$$\mathrm{Tree}_{\Gamma_f}(f^{t+\ell y}(r)) = \mathfrak{T}_{n'_{i_t}(\vec{\nu})},$$

where $n'_{i_t}(\vec{\nu})$ is the unique $n \in \{0, 1, \ldots, N\}$ such that $\vec{\nu}^{+} \in S_{i_t,n}$.

We remind the reader that we wish to output a compact description of the cyclic sequence of rooted tree isomorphism types that characterizes the connected component of Γ_f containing r. In order to do so, we compute sets $\overline{S}_{n,i_t}$ for $n \in \{0, 1, \ldots, N\}$ (and our still fixed t) that are defined as follows. The elements of $\overline{S}_{n,i_t}$ are precisely those logical sign tuples $\vec{\nu} \in \{\emptyset, \neg\}^{I_{i_t}}$ for which $n'_{i_t}(\vec{\nu}) = n$.

Let us describe how to compute these sets $\overline{S}_{n,i_t}$. We go through the $N + 1 \in O(d\,2^{d^2\,\mathrm{mpe}(q-1)+d})$ values for n (see also the argument for $r = 0_{\mathbb{F}_q}$ at the beginning of this subsection), and for each n, we go through the $|S_{n,i_t,H_{i_t}}|$ logical sign tuples $\vec{\nu''}$ in $S_{n,i_t,H_{i_t}}$ (the last entry of S_{n,i_t}) in their listed order, which is lexicographic. For each $\vec{\nu''}$, we set $\vec{\nu'} := \vec{\nu''} \diamond \vec{\xi}_{i_t,H_{i_t}} = (\nu'_j)_{j=1,2,\ldots,m_{i_t}}$. We check whether $\nu'_j = \nu_j$ for all $j \in I_{i_t}$ (which takes $O(m_{i_t} \log q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log q)$ bit operations). If not, we move on to the next $\vec{\nu''}$ and associated $\vec{\nu'}$, otherwise we add $\vec{\nu'}^{-}$ to $\overline{S}_{n,i_t}$ as a new element (again taking $O(m_{i_t} \log q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log q)$ bit operations, for copying). Because $N \in O(d\,2^{d^2\,\mathrm{mpe}(q-1)+d})$ and $\sum_{n=0}^{N} |S_{n,i_t,H_{i_t}}| \leq 2^{(H_{i_t}+1)d} \leq 2^{d^2\,\mathrm{mpe}(q-1)+d}$, and we only need $O(1)$ bit operations to deal with an n for which $S_{n,i_t} = \emptyset$, we conclude that the overall bit operation cost of computing the sets $\overline{S}_{n,i_t}$ (for our fixed t) is in

$$O(d\,2^{d^2\,\mathrm{mpe}(q-1)+d} + m_{i_t}\,2^{(H_{i_t}+1)d}\log q) \subseteq O(d^2\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q),$$

and the constructed arrays representing those sets are lexicographically ordered.

Let us now finally "unfix" t again. For all the $O(d)$ values of $t \in \{0, 1, \ldots, \ell-1\}$ together, a q-bounded query complexity of all computations we described after agreeing to fix t is

$$(d \log^{3+o(1)} q + d^3 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^3 \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} \log q, \\ 0, d^3 \operatorname{mpe}(q-1), d^3 \operatorname{mpe}(q-1), 0). \tag{5.4}$$

Here, the first summand in the bit operation component, as well as the specified queries, cover the cost of everything except the computation of the sets $\overline{S}_{n,i_t}$ themselves, which is instead covered by the second summand in the bit operation component (without needing any queries). Considering the query complexities of all other involved computations (see the detailed steps of the algorithm below), we find that only the second entry, 0, in formula (5.4) needs to be replaced by d in order to obtain a valid q-bounded query complexity of the entire algorithm for solving Problem 3.

As in the previous two subsections, we conclude with some pseudocode for this algorithm, which includes q-bounded query complexities for each step.

1 If $r = 0_{\mathbb{F}_q}$, then search for the unique $n \in \mathcal{N} = \{0, 1, \ldots, N\}$ such that $S_{n,d} = \emptyset$, and output the following and halt: "The cyclic sequence of rooted tree isomorphism types which encodes the digraph isomorphism type of the connected component in question is $[\mathfrak{D}_n]$."
QC: $(d 2^{d^2 \operatorname{mpe}(q-1)+d} \log q, 0, 0, 0, 0)$.

2 Compute the induced function $\bar{f}$ on $\{0, 1, \ldots, d\}$ and the affine maps A_i of $\mathbb{Z}/s\mathbb{Z}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

3 Let ω be the "natural" choice of primitive element of $\mathbb{F}_q$ (see the proof of Lemma 5.1.6 (4)), and determine $\mathfrak{l} := \log_\omega(r)$.
QC: $(\log q, 1, 0, 0, 0)$.

4 Set $i := \mathfrak{l} \bmod d$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

5 Determine the cycle $(i_0, i_1, \ldots, i_{\ell-1})$ of $i = i_0$ under $\bar{f}$.
QC: $(d \log d, 0, 0, 0, 0)$.

6 Compute $r'_{i_0} := (\mathfrak{l} - i)/d$ and $r'_{i_t} := A_{i_{t-1}}(r'_{i_{t-1}})$ for $t = 1, 2, \ldots, \ell - 1$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

7 For $t = 0, 1, \ldots, \ell - 1$, compute $\mathcal{A}_{i_t} := A_{i_t} A_{i_{(t+1) \bmod \ell}} \cdots A_{i_{(t+\ell-1) \bmod \ell}} : x \mapsto \overline{\alpha}_{i_t} x + \overline{\beta}_{i_t}$.
QC: $(d^2 \log^{1+o(1)} q, 0, 0, 0, 0)$.

8 For each $t = 0, 1, \ldots, \ell - 1$, do the following. QC:

$$(d \log^{3+o(1)} q + d^3 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^3 \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} \log q, \\ 0, d^3 \operatorname{mpe}(q-1), d^3 \operatorname{mpe}(q-1), 0).$$

8.1 Compute $\gcd(\overline{\alpha}_{i_t}, s)$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

8.2 Find the smallest non-negative integer L_{i_t} such that

$$\gcd(\overline{\alpha}_{i_t}^{L_{i_t}}, s) = \gcd(\overline{\alpha}_{i_t}^{L_{i_t}+1}, s) =: s''_{i_t}.$$

QC: $(\log^{3+o(1)} q, 0, 0, 0, 0)$.

8.3 Compute $s'_{i_t} := s/s''_{i_t}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

8.4 Compute $\mathfrak{f}_{i_t} := \sum_{z=0}^{L_{i_t}-1} \overline{\alpha}_{i_t}^z \overline{\beta}_{i_t} \bmod s''_{i_t}$. In doing so, do *not* add up all the summands, but use the geometric sum formula

$$\sum_{z=0}^{L_{i_t}-1} \overline{\alpha}_{i_t}^z = \begin{cases} \frac{\overline{\alpha}_{i_t}^{L_{i_t}}-1}{\overline{\alpha}_{i_t}-1}, & \text{if } \overline{\alpha}_{i_t} \neq 1, \\ L_{i_t}\overline{\alpha}_{i_t}, & \text{if } \overline{\alpha}_{i_t} = 1, \end{cases}$$

and the fact that $L_{i_t} \in O(\log q)$.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

8.5 Read off $\mathcal{P}_{i_t} = \mathfrak{P}(x \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}} : j = 1, 2, \ldots, m_{i_t})$ from the given partition-tree register.
QC: $(d^2 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

8.6 For $j = 1, 2, \ldots, m_{i_t}$, do the following.
QC: $(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q, 0, d^2 \operatorname{mpe}(q-1), d^2 \operatorname{mpe}(q-1), 0)$.

8.6.1 Compute the affine discrete logarithm

$$\mathfrak{l}_{i_t,j} := \log_{\mathcal{A}_{i_t}}^{(\mathfrak{a}_{i_t,j})}(r'_{i_t}, \mathfrak{b}_{i_t,j}).$$

If $\mathfrak{l}_{i_t,j} = \infty$, set $\nu_j := \neg$, $\mathrm{Type}_{\mathrm{I}}(j) := \mathrm{true}$, $\mathrm{Type}_{\mathrm{II}}(j) := \mathrm{false}$ and

$$\mathrm{Type}_{\mathrm{III}}(j) := \mathrm{false}.$$

Otherwise, just set $\mathrm{Type}_{\mathrm{I}}(j) := \mathrm{false}$.
QC: $(\log^{1+o(1)} q, 0, 1, 0, 0)$.

8.6.2 If $\mathrm{Type}_{\mathrm{I}}(j) = \mathrm{false}$, compute the cycle length $l_{i_t,j}$ of r'_{i_t} under $\mathcal{A}_{i_t}$ modulo $\mathfrak{a}_{i_t,j}$. If $l_{i_t,j} = 1$, set $\nu_j := \emptyset$, $\mathrm{Type}_{\mathrm{II}}(j) := \mathrm{true}$ and $\mathrm{Type}_{\mathrm{III}}(j) := \mathrm{false}$. Otherwise, set $\mathrm{Type}_{\mathrm{II}}(j) := \mathrm{false}$ and $\mathrm{Type}_{\mathrm{III}}(j) := \mathrm{true}$.
QC: $(\log^{1+o(1)} q, 0, 0, 1, 0)$.

8.7 Set

$$\mathcal{P}^{(i_t)} := \mathfrak{P}(y \equiv l_{i_t,j} \pmod{\mathfrak{l}_{i_t,j}} : 1 \leq j \leq m_{i_t}, \mathrm{Type}_{\mathrm{III}}(j) = \mathrm{true}).$$

QC: $(d^2 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

8.8 For each $n \in \mathcal{N} = \{0, 1, \ldots, N\}$, do the following.
QC: $(d^2 \operatorname{mpe}(q-1)2^{d^2 \operatorname{mpe}(q-1)+d} \log q, 0, 0, 0, 0)$.

8.8.1 Initialize the set $\overline{S}_{n,i_t}$ to be $\emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

8.8.2 For each $\vec{\nu}'' \in S_{n,i_t,H_{i_t}}$ (last entry of S_{n,i_t} from the partition-tree register from the input), do the following.
QC: $(d^2 \operatorname{mpe}(q-1)2^{d^2 \operatorname{mpe}(q-1)+d} \log q, 0, 0, 0, 0)$.

8.8.2.1 Set $\vec{\nu}' := \vec{\nu}'' \diamond \vec{\xi}_{i_t,H_{i_t}} = (\nu'_j)_{j=1,2,\ldots,m_{i_t}}$.
QC: $(d^2 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

8.8.2.2 Check whether $\nu'_j = \nu_j$ for all $j \in \{1, 2, \ldots, m_{i_t}\}$ such that one of $\text{Type}_{\text{I}}(j)$, $\text{Type}_{\text{II}}(j)$ or $\text{Type}_{\text{III}}(j)$ is true. If so, add $\vec{\nu}'^{-}$ (which is $\vec{\nu}'$ with all components corresponding to one of the three types deleted) to $\overline{S}_{n,i_t}$ as a new element.
QC: $(d^2 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

9 Output the following: "Consider an iterate $f^z(r)$, where $z \in \mathbb{Z}/l\mathbb{Z}$. Let $t := z \bmod \ell$ and $y := (z-t)/\ell \in \mathbb{Z}/(l/\ell)\mathbb{Z}$. Depending on t, we have the arithmetic partition $\mathcal{P}^{(i_t)}$ of $\mathbb{Z}/(l/\ell)\mathbb{Z}$, and the block of $\mathcal{P}^{(i_t)}$ in which y is contained controls the isomorphism type of $\text{Tree}_{\Gamma_f}(f^z(r))$. More precisely, for each $n \in \mathcal{N}$, we have the set $\overline{S}_{n,i_t}$ of logical sign tuples such that $\text{Tree}_{\Gamma_f}(f^z(r)) = \mathfrak{T}_n$ if and only if $y \in \mathcal{B}(\mathcal{P}^{(i_t)}, \vec{\nu})$ for some $\vec{\nu} \in \overline{S}_{n,i_t}$. The arithmetic partitions $\mathcal{P}^{(i_t)}$ and associated logical sign tuple sets $\overline{S}_{n,i_t}$ are as follows.", followed by printing (the computed spanning congruence sequence of) $\mathcal{P}^{(i_t)}$ and the sets $\overline{S}_{n,i_t}$ for each $t = 0, 1, \ldots, \ell - 1$. Then halt.
QC: $(d^3 \operatorname{mpe}(q-1)2^{d^2 \operatorname{mpe}(q-1)+d} \log q, 0, 0, 0, 0)$.

5.3 The isomorphism problem for functional graphs of generalized cyclotomic mappings

In this section, we consider the algorithmic problem of deciding whether the functional graphs of two given generalized cyclotomic mappings of $\mathbb{F}_q$ (each of a fixed index, but not necessarily both of the same index) are isomorphic digraphs. We note that thanks to Babai [9] (see also Helfgott's expository article [32], or its English translation [31]), a general algorithm for deciding the isomorphism of (di)graphs is known that is quasi-polynomial in the number of vertices. Assuming that the graphs in question are given by specifying their edges (as ordered or unordered pairs of vertices) individually, this algorithm is quasi-polynomial in the input length (one can assume without loss of generality that there are no isolated vertices, and thus that the number of vertices and the number of edges are within quadratic bounds of each other).

However, we do not specify our functional graphs Γ_f in this form; rather, we specify f in its cyclotomic form (1.1), so our input length is only in $O(d \log q)$, while the vertex number is $|\mathbb{F}_q| = q$. We believe that it is a hard problem to decide whether the functional graphs of two generalized cyclotomic mappings of $\mathbb{F}_q$, each of an index that is at most d, are isomorphic (i.e., that this problem is not generally solvable in polynomial time in $\log q$ for fixed d). However, for some special cases, efficient algorithms can be developed, and it is the purpose of this section to present such special cases and the associated decision algorithms.

5.3.1 Special case: Index 1

Generalized cyclotomic mappings of $\mathbb{F}_q$ of index 1 are the same as monomial mappings $x \mapsto ax^r$, where $a \in \mathbb{F}_q$ and $r \in \{1, \dots, q-1\}$ (if one has $r_0 = 0$ in formula (1.1) with $d = 1$, then one may replace it by $q-1$ to get a formula that works on all of $\mathbb{F}_q$). To get the class of all monomial mappings of $\mathbb{F}_q$, one must include the case "$r = 0$" (in which $0_{\mathbb{F}_q}$ is not necessarily fixed), and we do so in this subsection. Thanks to earlier work of Deng [21], it is easy to decide whether two monomial mappings of $\mathbb{F}_q$, say $f : x \mapsto ax^r$ and $f' : x \mapsto a'x^{r'}$, have isomorphic functional graphs, as we explain below.

First, we note that if f is constant, which happens if and only if $r = 0$ or $a = 0_{\mathbb{F}_q}$, then $\Gamma_f \cong \Gamma_{f'}$ if and only if f' is constant as well, i.e., if and only if $r' = 0$ or $a' = 0_{\mathbb{F}_q}$. Checking whether this special case applies only takes $O(\log q)$ bit operations (for scanning the values of a, a', r, r').

We may thus assume that r, r', a, a' all are non-zero. Because $a, a' \neq 0_{\mathbb{F}_q}$, we have $f(\mathbb{F}_q^*) \cup f'(\mathbb{F}_q^*) \subseteq \mathbb{F}_q^*$. Following the procedure described in our introduction, we can associate with f, respectively, f', an affine map A, respectively, A', of $\mathbb{Z}/(q-1)\mathbb{Z}$ such that $\Gamma_f \cong \Gamma_{f'}$ if and only if $\Gamma_A \cong \Gamma_{A'}$. Computing A and A' has q-bounded query complexity $(\log^{1+o(1)} q, 1, 0, 0, 0)$; beside some simple arithmetic, it requires the computation of the discrete logarithms of a and a' with base ω (the "natural choice" of primitive element of $\mathbb{F}_q$ – see the proof of Lemma 5.1.6 (4)).

In order to decide whether $\Gamma_A \cong \Gamma_{A'}$, we use the following result, which is a variant of [21, Theorem 11].

Theorem 5.3.1.1. *Let $m = p_1^{v_1} \cdots p_K^{v_K}$ be a positive integer with its prime factorization displayed. Let $a, a', b, b' \in \mathbb{Z}/m\mathbb{Z}$, and let us denote by A, respectively, A', the affine map $x \mapsto ax + b$, respectively, $x \mapsto a'x + b'$, of $\mathbb{Z}/m\mathbb{Z}$. Moreover, we define*

- *$\mathfrak{Y} := \{p_j : j \in \{1, 2, \dots, K\}, p_j \nmid a, A \bmod p_j^{v_j}$ has a fixed point$\}$, and analogously for $\mathfrak{Y}'$, with a' and A' in place of a and A; and*
- *l, respectively, l', to be the minimal cycle length of A, respectively, A', on its respective periodic points.*

Then $\Gamma_A \cong \Gamma_{A'}$ *if and only if both of the following hold:*

(1) $\gcd(a, m) = \gcd(a', m)$, *and*

(2) $\mathfrak{Y} = \mathfrak{Y}'$, $l = l'$, $\operatorname{ord}_{p^{\nu_p(m)}}(a^l) = \operatorname{ord}_{p^{\nu_p(m)}}((a')^l)$ *for all* $p \in \mathfrak{Y}$, *and if* $2 \in \mathfrak{Y}$ *and* $\nu_2(m) > 1$, *then* $\operatorname{ord}_4(a^l) = \operatorname{ord}_4((a')^l)$.

Proof. This is the same as [21, Theorem 11] except that in condition (2), we do not demand that $\operatorname{ord}_t(a^l) = \operatorname{ord}_t((a')^l)$ for *all* divisors t of m whose prime divisors are in $\mathfrak{Y}$. However, our condition (2) is enough, because if it holds and $t = \prod_{p\in\mathfrak{Y}} p^{\nu'_p}$ divides m, then for all $p \in \mathfrak{Y}$, one has

$$\operatorname{ord}_{p^{\nu'_p}}(a^l) = \operatorname{ord}_{p^{\nu'_p}}((a')^l), \tag{5.5}$$

and thus

$$\operatorname{ord}_t(a^l) = \operatorname{lcm}_{p\in\mathfrak{Y}} \operatorname{ord}_{p^{\nu'_p}}(a^l) = \operatorname{lcm}_{p\in\mathfrak{Y}} \operatorname{ord}_{p^{\nu'_p}}((a')^l) = \operatorname{ord}_t((a')^l).$$

To see that formula (5.5) holds, we make a case distinction.

- If $p > 2$, let us fix a primitive root r of $\mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$. Because

  $$\operatorname{ord}_{p^{\nu_p(m)}}(a^l) = \operatorname{ord}_{p^{\nu_p(m)}}((a')^l),$$

  it follows that modulo $p^{\nu_p(m)}$, we have $a^l = \mathrm{r}^k$ and $(a')^l = \mathrm{r}^{k'}$, where

  $$\gcd(k, \phi(p^{\nu_p(m)})) = \gcd(k', \phi(p^{\nu_p(m)})).$$

  Since $\phi(p^{\nu'_p})$ divides $\phi(p^{\nu_p(m)})$, it follows that

  $$\gcd(k, \phi(p^{\nu'_p})) = \gcd(k', \phi(p^{\nu'_p})),$$

  whence a^l and $(a')^l$ are also of the same multiplicative order modulo $p^{\nu'_p}$, as required.

- If $p = 2$, then formula (5.5) holds by assumption if $\nu_2(m) \le 2$, so let us assume that $\nu_2(m) \ge 3$. We may also assume that $\nu'_2 < \nu_2(m)$, because there is nothing to show if $\nu'_2 = \nu_2(m)$. By the structure of $\mathbb{Z}/2^{\nu_2(m)}\mathbb{Z}$, modulo $2^{\nu_2(m)}$ we can write $a^l = (-1)^{\varepsilon}5^k$ and $(a')^l = (-1)^{\varepsilon'}5^{k'}$ with $\varepsilon, \varepsilon' \in \{0, 1\}$. Since a^l and $(a')^l$ have (by assumption) the same multiplicative order modulo 4, we infer that $\varepsilon = \varepsilon'$. It is not hard to see that if $\operatorname{ord}_{2^{\nu_2(m)}}(a^l) = \operatorname{ord}_{2^{\nu_2(m)}}((a')^l) \in \{1, 2\}$, then $\operatorname{ord}_{2^{\nu'_2}}(a^l) = \operatorname{ord}_{2^{\nu'_2}}((a')^l) = 1$. Indeed, this is clear if both orders modulo $2^{\nu_2(m)}$ are equal to 1, and if both orders modulo $2^{\nu_2(m)}$ are equal to 2, then $\operatorname{ord}_{2^{\nu_2(m)}}(5^k), \operatorname{ord}_{2^{\nu_2(m)}}(5^{k'})$ both divide 2, whence due to $\nu'_2 < \nu_2(m)$, one has $5^k \equiv 5^{k'} \equiv 1 \pmod{2^{\nu'_2}}$, as follows by comparing the multiplicative orders of 5

modulo $2^{v_2(m)}$ and modulo $2^{v_2'}$, respectively. We may thus assume that the common multiplicative order modulo $2^{v_2(m)}$ of a^l and $(a')^l$ is strictly greater than 2. Then

$$\operatorname{ord}_{2^{v_2(m)}}(5^k) = \operatorname{ord}_{2^{v_2(m)}}(a^l) = \operatorname{ord}_{2^{v_2(m)}}((a')^l) = \operatorname{ord}_{2^{v_2(m)}}(5^{k'}),$$

and with an analogous argument to the one for "$p > 2$", we conclude that

$$\operatorname{ord}_{2^{v_2'}}(5^k) = \operatorname{ord}_{2^{v_2'}}(5^{k'}).$$

Therefore,

$$\begin{aligned}\operatorname{ord}_{2^{v_2'}}(a^l) &= \operatorname{lcm}(\operatorname{ord}_{2^{v_2'}}((-1)^\varepsilon), \operatorname{ord}_{2^{v_2'}}(5^k)) \\ &= \operatorname{lcm}(\operatorname{ord}_{2^{v_2'}}((-1)^\varepsilon), \operatorname{ord}_{2^{v_2'}}(5^{k'})) \\ &= \operatorname{ord}_{2^{v_2'}}((a')^l),\end{aligned}$$

as required. ∎

Using Theorem 5.3.1.1, we can prove the following.

Corollary 5.3.1.2. *Let m be a positive integer, and let A and A' be affine maps of $\mathbb{Z}/m\mathbb{Z}$. Deciding whether $\Gamma_A \cong \Gamma_{A'}$ takes m-bounded query complexity*

$$(\log^{2+o(1)} m, 0, \log m, 1, 0)$$

and m-bounded Las Vegas dual complexity

$$(\log^{8+o(1)} m, \log^{4+o(1)} m, \log^2 m).$$

Proof. We use the notation from Theorem 5.3.1.1, including that $A(x) = ax + b$ and $A'(x) = a'x + b'$. We argue that the conditions given in Theorem 5.3.1.1 can be verified within the specified m-bounded query complexity (the asserted Las Vegas dual complexity then follows readily using Lemma 5.1.7). First, we compute and compare $\gcd(a, m)$ and $\gcd(a', m)$, which takes $O(\log^{1+o(1)} m)$ bit operations by Lemma 5.1.5 (8). Next, we factor $m = p_1^{v_1} \cdots p_K^{v_K}$ using a single m-bounded mdl query (see also the beginning of Section 5.2).

Following that, we compute the sets $\mathfrak{Y}$ and $\mathfrak{Y}'$ and compare them. For this, we observe that $A \bmod p_j^{v_j}$, respectively, $A' \bmod p_j^{v_j}$, has a fixed point if and only if the congruence $(a-1)x \equiv -b \pmod{p_j^{v_j}}$, respectively, $(a'-1)x \equiv -b' \pmod{p_j^{v_j}}$, is solvable, which holds if and only if $\gcd(a-1, p_j^{v_j}) \mid b$, respectively, $\gcd(a'-1, p_j^{v_j}) \mid b'$. To check whether this holds, we read off the binary representation of $p_j^{v_j}$ from the output of the above mdl query, then carry out the relevant arithmetic in either case, which takes $O(\log^{1+o(1)} m)$ bit operations for a single j by Lemma 5.1.5 (1,3,8).

Because there are $O(\log m)$ distinct values of j, it takes $O(\log^{2+o(1)} m)$ bit operations altogether to compute $\mathfrak{Y}$ and $\mathfrak{Y}'$ and check whether they are equal.

If $\mathfrak{Y} = \mathfrak{Y}'$, we next compute the minimal cycle lengths l and l' and check if they are equal. It follows from our Table 2.2 (or [15, Tables 3 and 4], in which cycle types, not CRL-lists, of affine maps of finite primary cyclic groups are displayed and from which the cycle lengths can be read off more directly) that modulo a prime power, the cycle lengths of an affine permutation are linearly ordered under divisibility. Therefore, l, respectively, l', is the least common multiple of the smallest cycle lengths of A, respectively of A', modulo the $p_j^{v_j}$ for those $j \in \{1, 2, \dots, K\}$ such that $p_j \nmid a$, respectively, $p_j \nmid a'$. Those minimal cycle lengths can be computed according to our Table 2.2 (or [15, Tables 3 and 4]). More specifically, we go through $j = 1, 2, \dots, K$, and for each of these values, we do the following.

- We check whether $p_j \nmid a$, respectively, $p_j \nmid a'$, taking $O(\log^{1+o(1)} m)$ bit operations.
- If so, we check which case in Table 2.2 (or [15, Table 3 or 4, respectively]) applies, which also takes $O(\log^{1+o(1)} m)$ bit operations.
- Finally, we compute the minimal cycle length according to the case-specific formula and factor it, using one mdl query and $O(\log^{1+o(1)} m)$ bit operations.

For all j together, this process takes m-bounded query complexity

$$(\log^{2+o(1)} m, 0, \log m, 0, 0).$$

Following this, we determine the least common multiple of the cycle lengths. These lengths are already factored, so one only needs to go through the $O(\log m)$ primes dividing at least one of those cycle lengths (we note that each of those primes is a divisor of $p(p-1)$ for some prime $p \mid m$), and for each of them, we compute the largest exponent with which it occurs. For this, we need to scan the obtained factorization of the minimal cycle length modulo $p_j^{v_j}$, which is a bit string of length in $O(\log p_j^{v_j})$, for each j, and we need to compare the stored intermediate maximum with one of the prime exponents in it, which is a bit string of length in $O(\log\log p_j^{v_j})$. Altogether, the computation of l and l' takes $O(\log^{2+o(1)} m)$ bit operations, and checking whether l and l' are equal takes a mere $O(\log m)$ bit operations.

If $l = l'$, we next compute $a^l \bmod m$ and $(a')^l \bmod m$, taking $O(\log^{2+o(1)} m)$ bit operations. Then, for each $p \in \mathfrak{Y}$, we compute $\mathrm{ord}_{p^{v_p(m)}}(a^l)$ and $\mathrm{ord}_{p^{v_p(m)}}((a')^l)$, which can be done for all p together with just two mord queries, and check if they are equal, which takes $O(\log m)$ bit operations for all p together. Finally, if necessary, we compute $\mathrm{ord}_4(a^l)$ and $\mathrm{ord}_4((a')^l)$ and check if they are equal, which can be done with $O(\log^{1+o(1)} m)$ bit operations (no queries necessary). ∎

With regard to our application to generalized cyclotomic mappings of degree 1, we note the following consequence of Corollary 5.3.1.2.

Corollary 5.3.1.3. *Let f and f' be monomial mappings of $\mathbb{F}_q$, given in polynomial form. Within q-bounded query complexity*

$$(\log^{2+o(1)} q, 1, \log q, 1, 0),$$

or q-bounded Las Vegas dual complexity

$$(\log^{8+o(1)} q, \log^{4+o(1)} q, \log^2 q)$$

one can decide whether $\Gamma_f \cong \Gamma_{f'}$.

Proof. As explained at the beginning of this subsection, one first deals with the case where at least one of f or f' is constant through simple scans of the input, taking $O(\log q)$ bit operations. If not, then

$$f(x) = ax^r \quad \text{and} \quad f'(x) = a'x^{r'},$$

where a, a', r, r' all are non-zero. Under logarithmization, the restriction of f, respectively, f', to $\mathbb{F}_q^*$ corresponds to the affine map

$$A(x) = rx + \log_\omega(a),$$

respectively, $A'(x) = r'x + \log_\omega(a')$, of $\mathbb{Z}/(q-1)\mathbb{Z}$, and computing these affine maps takes q-bounded query complexity $(\log q, 1, 0, 0, 0)$. Finally, one applies the algorithm from the proof of Corollary 5.3.1.2 to check within q-bounded query complexity $(\log^{2+o(1)} q, 0, \log q, 1, 0)$ whether $\Gamma_A \cong \Gamma_{A'}$, which is equivalent to

$$\Gamma_f \cong \Gamma_{f'}.$$

■

Comparing our Theorem 5.3.1.1 with Deng's original version [21, Theorem 11], we note that the additional simplification of essentially only having to check the equality

$$\mathrm{ord}_t(a^l) = \mathrm{ord}_t((a')^l)$$

for those divisors t of m that are of the form $p^{v_p(m)}$, as opposed to all divisors of m, is essential to achieve a polynomial complexity in the associated algorithm. This is because the number $\tau(m)$ of distinct (positive) divisors of m may be superpolynomial in $\log m$.

On the other hand, Dirichlet proved a result which implies that the average value of τ over the initial segment $\{1, 2, \ldots, N\}$ of $\mathbb{N}^+$ is asymptotically equivalent to $\log N$ [8, Theorem 3.3], so in particular, the set of positive integers m for which $\tau(m) \geq \log^{1+\varepsilon} m$ has asymptotic density 0 for each constant $\varepsilon > 0$, because otherwise,

if the said asymptotic density is $\delta > 0$, there are infinitely many $N \in \mathbb{N}^+$ such that

$$\begin{aligned} N^{-1} \sum_{m \le N} \tau(m) &\ge N^{-1} \sum_{m=1}^{\lfloor \delta N \rfloor} \log^{1+\varepsilon} m \\ &\ge (2N)^{-1} \frac{\delta}{2} N \log^{1+\varepsilon} \left(\frac{\delta}{2} N \right) = \frac{\delta}{4} \log^{1+\varepsilon} \left(\frac{\delta}{2} N \right) \\ &\ge \frac{\delta}{8} \log^{1+\varepsilon} N \gg \log N, \end{aligned}$$

a contradiction. Concerning the average value of $\tau(q-1)$, where q ranges over prime powers, we have the following result, the "In particular" statement of which will be used in Section 5.3.2. In this context, the authors would like to thank Ofir Gorodetsky, who kindly pointed out Halberstam's crucial paper [27] and parts of the proof of Proposition 5.3.1.4 in an answer to a question posted by the first author on MathOverflow[2].

Proposition 5.3.1.4. *For each $\varepsilon > 0$, there are constants $c_\varepsilon, c'_\varepsilon > 0$ such that the following hold for all but an asymptotic fraction of less than ε of all prime powers q:*

(1) $\mathrm{mpe}(q-1) < c_\varepsilon$; *and*

(2) *the number of distinct prime divisors of $q-1$ is less than $c'_\varepsilon \log\log q$.*

In particular, for all such prime powers q, one has $\tau(q-1) < \log^{c''_\varepsilon} q$, where

$$c''_\varepsilon := \log(c_\varepsilon + 1) c'_\varepsilon.$$

Proof. Throughout this proof, the variable q ranges over prime powers, while p ranges over primes. Statement (1) is the same as Proposition 5.1.10 (1). For statement (2), as usual, we denote by $\omega(m)$ the number of distinct prime divisors of $m \in \mathbb{N}^+$. Halberstam proved that

$$\Big(\sum_{p \le x} 1 \Big)^{-1} \sum_{p \le x} \omega(p-1) \sim \log\log x \tag{5.6}$$

as $x \to \infty$, see [27, Theorem 1]. Now, let $\varepsilon > 0$ be fixed, and let us assume that for some constant $c > 0$, one has $\omega(q-1) \ge c \log\log q$ for an asymptotic fraction of at least ε of all prime powers q. We need to bound c in terms of ε in order to prove statement (2). Now, because proper prime powers are a density 0 subset of all prime powers (see the proof of Proposition 5.1.10), we conclude that also for an asymptotic fraction of at least ε of all *primes* p, one has $\omega(p-1) \ge c \log\log p$. We fix a large

[2] see https://mathoverflow.net/questions/436134/average-value-of-the-prime-omega-function-omega-on-predecessors-of-prime-powe, visited on 2 September 2025.

enough $x \geq 2$ such that for a fraction of at least $\varepsilon/2$ of all primes $p \leq x$, one has $\omega(p-1) \geq c \log\log p$. Let us denote by p_m the m-th prime number for $m \in \mathbb{N}^+$. Because $p_m \geq (m \log m)/2$ for large enough m, and the total number of primes $p \leq x$ is at least $x/(2\log x)$, it follows that

$$\sum_{p \leq x} \omega(p-1) \geq \sum_{m=1}^{\lfloor \varepsilon x/(4\log x)\rfloor} c \log\log p_m \geq \sum_{m=\lceil \varepsilon x/(8\log x)\rceil}^{\lfloor \varepsilon x/(4\log x)\rfloor} c \log\log\left(\frac{1}{2} m \log m\right)$$
$$\geq \frac{\varepsilon x}{16 \log x} \cdot c \log\log\left(\frac{1}{2} \cdot \frac{\varepsilon x}{8\log x} \cdot \log\left(\frac{\varepsilon x}{8\log x}\right)\right),$$

whence

$$\left(\sum_{p \leq x} 1\right)^{-1} \sum_{p \leq x} \omega(p-1)$$
$$\geq \left(2\frac{x}{\log x}\right)^{-1} \cdot \frac{\varepsilon x}{16 \log x} \cdot c \log\log\left(\frac{1}{2} \cdot \frac{\varepsilon x}{8\log x} \cdot \log\left(\frac{\varepsilon x}{8\log x}\right)\right)$$
$$\sim \frac{\varepsilon c}{32} \log\log x,$$

which implies

$$\left(\sum_{p \leq x} 1\right)^{-1} \sum_{p \leq x} \omega(p-1) \geq \frac{\varepsilon c}{64} \log\log x$$

if x is large enough. Hence, in order to not contradict Halberstam's (5.6), we must have $c \leq 64/\varepsilon$, an upper bound on c in terms of ε, as required in order for statement (2) to hold.

Finally, for the "In particular" statement, we note that because

$$\tau(q-1) = (v_1+1)(v_2+1)\cdots(v_K+1)$$

if $q - 1 = p_1^{v_1} p_2^{v_2} \cdots p_K^{v_K}$ is the prime factorization of $q-1$, one can bound $\tau(q-1)$ from above as follows:

$$\tau(q-1) \leq (\mathrm{mpe}(q-1)+1)^{\omega(q-1)} \leq (c_\varepsilon + 1)^{c'_\varepsilon \log\log q}$$
$$= \exp(c'_\varepsilon \log\log q \log(c_\varepsilon + 1))$$
$$= (\log q)^{\log(c_\varepsilon+1)c'_\varepsilon}.$$

■

In addition to thanking Ofir Gorodetsky, the authors would also like to thank one of the anonymous reviewers, who pointed out that it is even possible to bound the average value of $\tau(q-1)$ for prime powers $q \leq x$ using the Titchmarsh divisor theorem, which states that for each constant $a \in \mathbb{Z}$, one has

$$\sum_{p \leq x} \tau(p+a) = C_1(a) + O\left(\frac{x \log\log x}{\log x}\right),$$

see, e.g., Halberstam's short proof [28] (using the Bombieri–Vinogradov theorem and the Brun–Titchmarsh inequality) or the earlier proof of Linnik [46] using the dispersion method. Indeed, using that $\sum_{p\le x} 1 \sim \frac{x}{\log x}$, this implies that

$$\Big(\sum_{p\le x} 1\Big)^{-1} \sum_{p\le x} \tau(p-1) \in O(\log x),$$

and since the number of proper prime powers up to x is in $O(\sqrt{x}\log x)$ and τ grows asymptotically more slowly than any power function, the same O-bound holds true when primes p are replaced by prime powers q. In particular, we get the following stronger version of the "In particular" of Proposition 5.3.1.4.

Proposition 5.3.1.5. *Let $\mathfrak{g} : [0,\infty) \to \mathbb{R}$ be monotonically increasing and such that $\mathfrak{g}(x) \to \infty$ as $x \to \infty$. Then, as $x \to \infty$, the proportion of prime powers $q \le x$ such that $\tau(q-1) \ge \mathfrak{g}(q-1)$ tends to 0.*

Proof. Assume otherwise. Then there is an $\varepsilon > 0$ such that for arbitrarily large $x \in [0,\infty)$, the proportion of prime powers $q \le x$ with $\tau(q-1) \ge \mathfrak{g}(x)$ is at least ε. Since at least $\frac{\varepsilon}{2}x$ of those prime powers must be greater than or equal to $\frac{\varepsilon}{2}x$, it follows that there is are arbitrarily large positive real numbers x such that

$$\Big(\sum_{q\le x} 1\Big)^{-1} \sum_{q\le x} \tau(q-1) \ge \frac{\mathfrak{g}\big(\frac{\varepsilon}{2}x\big)\cdot\frac{\varepsilon}{2}x}{2x/\log x} = \frac{\varepsilon}{2}\log x \cdot \mathfrak{g}\Big(\frac{\varepsilon}{2}x\Big),$$

contradicting that the average value of $\tau(q-1)$ is in $O(\log x)$. ∎

We conclude this subsection with two batches of pseudocode. First, we give pseudocode for the algorithm that checks whether $\Gamma_A \cong \Gamma_{A'}$, where $A : x \mapsto ax+b$ and $A' : x \mapsto a'x+b'$ are affine maps of $\mathbb{Z}/m\mathbb{Z}$ (see Corollary 5.3.1.2 and its proof). As in the previous section, we specify the (m-bounded) query complexity (QC) of each step.

1 Compute $\gcd(a,m)$ and $\gcd(a',m)$, and check whether they are equal. If not, output "false" and halt.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

2 Factor $m = p_1^{v_1}\cdots p_K^{v_k}$.
QC: $(\log m, 0, 1, 0, 0)$.

3 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(\log^{2+o(1)} m, 0, 0, 0, 0)$.

3.1 Check whether it is the case that $p_j \nmid a$ and $\gcd(a-1, p_j^{v_j}) \mid b$. If so, set $\mathrm{test}_j :=$ true, otherwise set $\mathrm{test}_j :=$ false.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

3.2 Check whether it is the case that $p_j \nmid a'$ and $\gcd(a'-1, p_j^{v_u}) \mid b'$. If so, set $\text{test}'_j :=$ true, otherwise set $\text{test}'_j :=$ false.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

3.3 If $\text{test}_j \neq \text{test}'_j$, then output "false" and halt.
QC: $(\log \log m, 0, 0, 0, 0)$.

4 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(\log^{2+o(1)} m, 0, \log m, 0, 0)$.

4.1 Check whether $p_j \nmid a$. If not, set $\text{Test}_j :=$ false and skip to step 4.4. Otherwise, set $\text{Test}_j :=$ true.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.2 Check which case in Table 2.2 (or [15, Table 3 or 4, respectively]) applies to $A \bmod p_j^{v_j}$, using simple arithmetic.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.3 Determine the minimal cycle length l_j of $A \bmod p_j^{v_j}$ according to Table 2.2 (or [15, Table 3 or 4, respectively]) and factor it.
QC: $(\log^{1+o(1)} m, 0, 1, 0, 0)$.

4.4 Check whether $p_j \nmid a'$. If not, set $\text{Test}'_j :=$ false and skip to the next j. Otherwise, set $\text{Test}'_j :=$ true.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.5 Check which case in Table 2.2 (or [15, Table 3 or 4, respectively]) applies to $A' \bmod p_j^{v_j}$, using simple arithmetic.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.6 Determine the minimal cycle length l'_j of $A' \bmod p_j^{v_j}$ according to Table 2.2 (or [15, Table 3 or 4, respectively]) and factor it.
QC: $(\log^{1+o(1)} m, 0, 1, 0, 0)$.

5 Compute

$$l := \operatorname{lcm}(l_j : 1 \leq j \leq K, \text{Test}_j = \text{true})$$

and

$$l' := \operatorname{lcm}(l'_j : 1 \leq j \leq K, \text{Test}'_j = \text{true})$$

using the factorizations of the l_j and l'_j computed in steps 4.3 and 4.6 above.
QC: $(\log^{2+o(1)} m, 0, 0, 0, 0)$.

6 Check whether $l = l'$. If not, output "false" and halt.
QC: $(\log m, 0, 0, 0, 0)$.

7 Compute $a^l \bmod m$ and $(a')^l \bmod m$.
QC: $(\log^{2+o(1)} m, 0, 0, 0, 0)$.

8 Compute $\mathrm{ord}_{p^{v_p(m)}}(a^l)$, respectively, $\mathrm{ord}_{p^{v_p(m)}}((a')^l)$, for all primes $p \mid m$ such that $p \nmid a$, respectively, $p \nmid a'$, in particular for all

$$p \in \mathfrak{Y} = \mathfrak{Y}' = \{p_j : \mathrm{test}_j = \mathrm{true}\}.$$

QC: $(\log m, 0, 0, 1, 0)$.

9 Check whether $\mathrm{ord}_{p^{v_p(m)}}(a^l) = \mathrm{ord}_{p^{v_p(m)}}((a')^l)$ for all $p \in \mathfrak{Y}$. If not, output "false" and halt.
QC: $(\log m, 0, 0, 0, 0)$.

10 If $4 \mid m$, do the following.

10.1 Check whether $\mathrm{ord}_4(a^l) = \mathrm{ord}_4((a')^l)$. If not, output "false" and halt.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

10.2 Output "true" and halt.
QC: $(1, 0, 0, 0, 0)$.

11 Else do the following.

11.1 Output "true" and halt.
QC: $(1, 0, 0, 0, 0)$.

Finally, we provide pseudocode for the algorithm that checks whether $\Gamma_f \cong \Gamma_{f'}$ for monomial mappings $f : x \mapsto ax^r$ and $f' : x \mapsto a'x^{r'}$ of $\mathbb{F}_q$ (where $a, a' \in \mathbb{F}_q$ and $r, r' \in \{0, 1, \ldots, q-1\}$).

1 If $a = 0_{\mathbb{F}_q}$ or $r = 0$, then do the following.

1.1 Check whether it is the case that $a' = 0_{\mathbb{F}_q}$ or $r' = 0$. If so, output "true" and halt. Otherwise, output "false" and halt.
QC: $(\log q, 0, 0, 0, 0)$.

2 Else do the following.

2.1 Check whether it is the case that $a' = 0_{\mathbb{F}_q}$ or $r' = 0$. If so, output "false" and halt.
QC: $(\log q, 0, 0, 0, 0)$.

2.2 Compute $b := \log_\omega(a)$ and $b' := \log_\omega(a')$, where ω is the "natural" choice of primitive element of $\mathbb{F}_q$ (see the proof of Lemma 5.1.6 (4)). Denote by A, respectively, A', the affine map of $\mathbb{Z}/(q-1)\mathbb{Z}$ given by the formula $A(x) = rx + b$, respectively, $A'(x) = r'x + b'$.
QC: $(\log q, 1, 0, 0, 0)$.

2.3 Use the algorithm from above to check whether

$$\Gamma_A \cong \Gamma_{A'}.$$

If so, output "true" and halt. Otherwise, output "false" and halt.
QC: $(\log^{2+o(1)} q, 0, \log q, 1, 0)$.

5.3.2 Special case: Trees only depend on the coset

In this subsection, we discuss two special classes of generalized cyclotomic mappings of $\mathbb{F}_q$ such that if f_1 and f_2 each belong to one of those classes (not necessarily both to the same) and are of index d_1 and d_2, respectively, then it can be decided whether $\Gamma_{f_1} \cong \Gamma_{f_2}$ in a q-bounded query complexity each entry of which is polynomial in the parameters $\max\{d_1, d_2\}$, $\log q$ and $\tau(q-1)$ (the number of divisors of $q-1$). In particular, the q-bounded Las Vegas dual complexity of this problem is always subexponential in the input size $O(\max\{d_1, d_2\} \log q)$, as $\tau(m) \in o(m^\varepsilon)$ for each $\varepsilon > 0$ [8, p. 296]. Moreover, in view of Proposition 5.3.1.4 or 5.3.1.5 with $\mathfrak{g}(x) = \log x$, for instance, for each $\varepsilon > 0$, one has that for all but an asymptotic fraction of less than ε of all finite fields $\mathbb{F}_q$, the said q-bounded Las Vegas dual complexity is polynomial in the input size (and the polynomial degree does, by Proposition 5.3.1.5, *not* depend on ε).

The two classes of generalized cyclotomic mappings f of $\mathbb{F}_q$, say of index d, which we consider are as follows.

- *Class 1*: f maps each coset C_i for $i \in \{0, 1, \ldots, d-1\}$ either to $C_d = \{0_{\mathbb{F}_q}\}$ or bijectively to $C_{\bar{f}(i)}$ (i.e., whenever the affine function A_i of $\mathbb{Z}/s\mathbb{Z}$ is well defined, it is a permutation of $\mathbb{Z}/s\mathbb{Z}$). If this happens, we say that f is *of special type* I. This is the same situation as in Section 4.3.
- *Class 2*: The induced function $\bar{f}$ is a permutation of $\{0, 1, \ldots, d\}$ (i.e., f permutes the cosets of C). If this happens, we say that f is *of special type* II. Using the notation of Section 3.2, this means that $\Gamma_f = \Gamma_{\mathrm{per}}$, and so the discussion from that section applies.

The crucial property which these two cases share is that the rooted trees above periodic vertices in Γ_f only depend on the block C_i, for $i \in \{0, 1, \ldots, d\}$, in which these vertices lie, as follows from Lemma 4.3.1 and Theorem 3.2.1 (or Proposition 2.1.8), respectively. This allows us to produce a comparatively compact description of the digraph isomorphism type of Γ_f. To that end, it is helpful to adapt the notion of a partition-tree register, introduced in Definition 5.1.2, as follows.

Definition 5.3.2.1. Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$.

(1) We assume that f is of special type I, so that *all* vertices (not just f-periodic ones) in a given block C_i have isomorphic rooted trees above them in Γ_f. A *type*-I *tree register for* f is an ordered sequence $((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ such that

   (a) the sets $\mathfrak{D}_n$ form a recursive tree description list, with associated rooted tree isomorphism types $\mathfrak{T}_n$ (see Definition 5.1.1) and

   (b) for each n, the set S_n is nonempty and consists precisely of those $i \in \{0, 1, \ldots, d\}$ such that $\mathrm{Tree}_{\Gamma_f}(x) \cong \mathrm{Tree}_{\Gamma_{\bar{f}}}(i) \cong \mathfrak{T}_n$ for any $x \in C_i$.

(2) We assume that f is of special type II, so that for each $i \in \{0, 1, \ldots, d-1\}$, the rooted tree isomorphism type $\mathrm{Tree}_{\Gamma_f}(x)$ for $x \in C_i$ only depends on i and the $\mathfrak{h}$-value of x (see Section 4.1, page 66 onward). A *type*-II *tree register for* f is an ordered sequence $((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ such that

(a) the sets $\mathfrak{D}_n$ form a recursive tree description list, with associated rooted tree isomorphism types $\mathfrak{T}_n$;

(b) the $\mathfrak{T}_n$ are just those isomorphism types that occur among the rooted trees of the form $\mathrm{Expand}(\mathfrak{T}_{i,h})$ for $i \in \{0, 1, \ldots, d-1\}$ and

$$h \in \{0, 1, \ldots, H_i\}$$

(see Section 4.1, page 66 onward, for the definition of the $\mathfrak{T}_{i,h}$); and

(c) for each n, one has $S_n = (\mathrm{ht}(\mathfrak{T}_n), S_{n,\mathrm{trans}}, S_{n,\mathrm{per}})$, where

$$S_{n,\mathrm{trans}} = \{i \in \{0, 1, \ldots, d-1\} : \mathrm{ht}(\mathfrak{T}_n) < H_i \text{ and } \mathfrak{T}_n = \mathrm{Expand}(\mathfrak{T}_{i,\mathrm{ht}(\mathfrak{T}_n)})\}$$

and

$$S_{n,\mathrm{per}} = \{i \in \{0, 1, \ldots, d-1\} : \mathfrak{T}_n = \mathrm{Expand}(\mathfrak{T}_{i,H_i})\}.$$

In an implementation, we assume that the sets S_n from Definition 5.3.2.1 (1), as well as the sets $S_{n,\mathrm{trans}}$ and $S_{n,\mathrm{per}}$ from Definition 5.3.2.1 (2), are represented by *sorted* arrays each entry of which is a binary digit representation of a number $i \in \{0, 1, \ldots, d\}$ with bit length exactly $\lfloor \log_2 d \rfloor + 1$. Moreover, in a type-I tree register, only one of the descriptions $\mathfrak{D}_n$ corresponds to $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, and it is the only description in which the second entries of elements may be larger than d. For the sake of efficiency, we make the convention that all descriptions $\mathfrak{D}_n$ except that one use $\lfloor \log_2 d \rfloor + 1$ digits for representing each entry m or k_m of an element (m, k_m) of $\mathfrak{D}_n$. On the other hand, in the description corresponding to $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, we use $\lfloor \log_2 d \rfloor + 1$ digits for the first entry m, and $\lfloor \log_2 q \rfloor + 1$ digits for the second entry k_m. In contrast to this, in a type-II tree register, we know that $m \leq d^2 \,\mathrm{mpe}(q-1) + d \leq d^2 \lfloor \log_2 q \rfloor + d$ (see the first paragraph in the proof of Lemma 5.3.2.2 (4) below), while there is a priori no upper bound on k_m other than the trivial one, q. Hence, in such a register, we use $\lfloor \log_2(d^2 \lfloor \log_2 q \rfloor + d) \rfloor + 1$ digits for representing m, and $\lfloor \log_2 q \rfloor + 1$ digits for representing k_m. In either case, the entries of a given description $\mathfrak{D}_n$ all have the same bit length, and different descriptions $\mathfrak{D}_n$ use the same bit length for the first entries of their elements. As before, we assume that the array representing a given description $\mathfrak{D}_n$ is lexicographically ordered (corresponding to the ordering of the elements (m, k_m) of $\mathfrak{D}_n$ by increasing m).

We observe that in Definition 5.3.2.1 (2,c), the set $S_{n,\mathrm{trans}}$ consists precisely of those $i \in \{0, 1, \ldots, d-1\}$ such that $\mathfrak{T}_n = \mathrm{Tree}_{\Gamma_f}(x)$ for all $x \in C_i$ with $\mathfrak{h}(x) = \mathrm{ht}(\mathfrak{T}_n) < H_i$, regardless of whether such x exist; we recall from Example 3.2.2 that

not necessarily all values in $\{0, 1, \ldots, H_i - 1\}$ are assumed by $\mathfrak{h}$ on the f-transient points in a given coset C_i. We also remind the reader that f-transient $x \in C_i$ are characterized by the inequality $\mathfrak{h}(x) < H_i$, and that for all such x, one has

$$\mathrm{ht}(\mathrm{Tree}_{\Gamma_f}(x)) = \mathrm{ht}(\mathfrak{T}_{i,\mathfrak{h}(x)}) = \mathfrak{h}(x);$$

see the recursive definition of the $\mathfrak{T}_{i,h}$ in Section 4.1, page 66 onward. Moreover, the set $S_{n,\mathrm{per}}$ consists of those $i \in \{0, 1, \ldots, d-1\}$ such that $\mathfrak{T}_n$ is the unique isomorphism type $\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Expand}(\mathfrak{T}_{i,H_i})$ for f-periodic $x \in C_i$ (and $\mathrm{Expand}(\mathfrak{T}_{i,H_i})$ is *not* necessarily of height H_i, but it is of height $\mathcal{H}_i$).

Next, we discuss the following important lemma.

Lemma 5.3.2.2. *Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$, given in cyclotomic form (1.1).*

(1) *Checking whether f is of special type* I *takes q-bounded query complexity*

$$(d \log^{1+o(1)} q, d, 0, 0, 0).$$

(2) *If f is of special type* I*, then a type-*I *tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d)$ can be computed within q-bounded query complexity*

$$(d^3 \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0).$$

(3) *Checking whether f is of special type* II *has q-bounded query complexity*

$$(d \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0).$$

(4) *If f is of special type* II*, then a type-*II *tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d^2 \,\mathrm{mpe}(q-1))$ can be computed within q-bounded query complexity*

$$(d^5 \log^2 d \,\mathrm{mpe}(q-1)^3 + d^5 \,\mathrm{mpe}(q-1)^3 \log q + d^2 \,\mathrm{mpe}(q-1) \log^{1+o(1)} q, d, 0, 0, 0).$$

In particular, it can be computed within q-bounded query complexity

$$(d^{5+o(1)} \log^4 q, d, 0, 0, 0).$$

Proof. For statement (1), we first compute $\bar{f}$ and the affine maps $A_i : x \mapsto \alpha_i x + \beta_i$, which requires q-bounded query complexity $(d \log^{1+o(1)} q, d, 0, 0, 0)$ by Proposition 5.1.8. We note that f is of special type I if and only if $\gcd(\alpha_i, s) = 1$ for all i such that A_i is well defined (i.e., such that the coefficient $a_i \in \mathbb{F}_q$ in the cyclotomic form (1.1) of f is non-zero), which one can check with $O(d \log^{1+o(1)} q)$ bit operations by Lemma 5.1.5 (3,8).

For statement (2), we start by computing $\bar{f}$, the number $s = (q-1)/d$ and (as in Section 5.2.1, page 116) the "layers"

$$\mathrm{Layer}_h := \begin{cases} \mathrm{im}(\bar{f}^h) \setminus \mathrm{im}(\bar{f}^{h+1}), & \text{if } h \in \{0, 1, \ldots, \bar{H}-1\}, \\ \mathrm{im}(\bar{f}^{\bar{H}}) = \mathrm{per}(\bar{f}), & \text{if } h = \infty, \end{cases}$$

of $\bar{f}$ with respect to iteration in sorted form and without multiple entries, where $\bar{H}$ is the maximum tree height in $\Gamma_{\bar{f}}$. Altogether, this takes q-bounded query complexity $(d \log^{1+o(1)} q + d \log^2 d, d, 0, 0, 0)$, and we may also compute the cycles of $\bar{f}$ in the process. After this, we start building the register. At any given point during that process, we have at least a "partial register" as an intermediate result, which includes definitions of descriptions $\mathfrak{D}_n$ of rooted trees $\mathfrak{T}_n$ for all $n \in \mathcal{N}$, an initial segment of $\mathbb{N}_0$. Because each $n \in \mathcal{N}$ has a non-empty subset S_n of $\{0, 1, \ldots, d\}$ associated with it and those sets are pairwise disjoint, we conclude that $|\mathcal{N}| \leq d + 1 \in O(d)$. In particular, $N \in O(d)$ in the end, as asserted.

Now, to build the register, we do the following: first, we compute the pre-image set $\bar{f}^{-1}(\{i\})$ in sorted form, which just requires looping over the known value table of $\bar{f}$ once, thus taking $O(d \log d)$ bit operations only. Now, for a fixed index $i \in \mathrm{Layer}_h$, let $j_1, j_2, \ldots, j_K$ be the $\bar{f}$-*transient* pre-images of i under $\bar{f}$; if $h < \infty$, those are simply all pre-images of i, and if $h = \infty$, one can determine them by additionally identifying the unique $\bar{f}$-pre-image of i in $\mathrm{per}(\bar{f}) = \mathrm{Layer}_\infty$; since the cycles of $\bar{f}$ are known and one can store the Boolean information which indices lie in Layer_∞ by scanning once over each of the other layers beforehand using $O(d \log d)$ bit operations, this only takes $O(\log d)$ bit operations per i, hence $O(d \log d)$ bit operations overall.

In what follows, we assume that i is fixed. Each j_t lies in a unique layer $\mathrm{Layer}_{h_{j_t}}$ with $h_{j_t} < h_i = h$, and so there is a unique non-negative integer $\bar{n}_{j_t} \in \mathcal{N}$ such that $j_t \in S_{\bar{n}_{j_t}}$. Computing $\bar{n}_{j_t}$ takes $O(|\mathcal{N}| \cdot \log^2 d) \subseteq O(d \log^2 d)$ bit operations for a single t, hence $O(d^2 \log^2 d)$ bit operations altogether (for this fixed value of i). Now, by Lemma 4.3.1, the rooted tree above any $x \in C_i$ is isomorphic to

$$\begin{cases} \mathrm{Tree}_{\Gamma_{\bar{f}}}(i) \cong \sum_{t=1}^{K} \mathfrak{T}^+_{\bar{n}_{j_t}}, & \text{if } i < d, \\ \sum_{t=1}^{K} s\mathfrak{T}^+_{\bar{n}_{j_t}}, & \text{if } i = d, \end{cases}$$

and so we may choose the following compact description $\mathfrak{D}$ for this tree.

- If $i < d$, we set

$$\mathfrak{D} := \{(n, m) : n \in \{\bar{n}_{j_t} : 1 \leq t \leq K\}, m = |\{t \in \{1, \ldots, K\} : \bar{n}_{j_t} = n\}| > 0\}.$$

- If $i = d$, we set

$$\mathfrak{D} := \{(n, sm) : n \in \{\bar{n}_{j_t} : 1 \leq t \leq K\}, m = |\{t \in \{1, \ldots, K\} : \bar{n}_{j_t} = n\}| > 0\}.$$

Computing $\mathfrak{D}$ after the numbers $\bar{n}_{j_t}$ have been determined requires us to create a list of the *distinct* values of the $\bar{n}_{j_t}$ and their multiplicities, which can be done in $O(K \log K \log d) \subseteq O(d \log^2 d)$ bit operations when using the sorting algorithm from Lemma 5.1.5 (10). If $i < d$, this is also the overall complexity of computing $\mathfrak{D}$ for that i, whereas if $i = d$, the complexity of computing $\mathfrak{D}$ is in $O(d \log^2 d + d \log^{1+o(1)} q)$, since it also involves integer multiplications. After $\mathfrak{D}$ has been computed, we check whether there is an $n \in \mathcal{N}$ such that $\mathfrak{D} = \mathfrak{D}_n$, which takes

$$O(|\mathcal{N}| \cdot d \log d) \subseteq O(d^2 \log d)$$

bit operations regardless of the value of i. Indeed, if $i = d$, for which the bit length of the second entries of elements of $\mathfrak{D}$ is not necessarily in $O(\log d)$, one can proceed as follows. Observing that those second entries can only be that large for this one value of i, one first checks whether $s > d$, which can be done with a mere $O(\log d)$ bit operations (we note that s itself was already computed at the beginning). If so, one knows that $\mathfrak{D} \neq \mathfrak{D}_n$ for any $n \in \mathcal{N}$; otherwise, the bit length of the second entries of $\mathfrak{D}$ is in $O(\log d^2) = O(\log d)$ even for $i = d$, and one can proceed as for $i < d$. In any case, if $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then we add i to S_n as a new element by merging the sorted lists corresponding to the sets S_n and $\{i\}$, which takes $O(d \log d)$ bit operations by Lemma 5.1.5 (11). Otherwise, we create $\mathfrak{D}$ as a new description $\mathfrak{D}_{n'}$, where $n' = \max \mathcal{N} + 1$, and initialize $S_{n'} := \{i\}$.

Since we need to carry out the computations described after declaring i to be fixed for all such i, the total bit operation cost of these computations is in $O(d^3 \log^2 d + d \log^{1+o(1)} q)$, and so the overall q-bounded query complexity of computing a type-I tree register for f is as asserted.

For statement (3), we simply compute $\bar{f}$ and check whether $\mathrm{im}(\bar{f}) = \{0, 1, \ldots, d\}$. The former of these two tasks takes q-bounded query complexity

$$(d \log^{1+o(1)} q, d, 0, 0, 0)$$

by Proposition 5.1.8, and the latter takes $O(d \log^2 d)$ bit operations (see the beginning of the argument in Section 5.2.1).

For statement (4), we first compute $\bar{f}$, the affine maps $A_i : x \mapsto \alpha_i x + \beta_i$ and (as in Section 5.2.1) a CRL-list $\bar{\mathcal{L}}$ of $\bar{f}$ and the cycles of $\bar{f}$, taking q-bounded query complexity in $(d \log^{1+o(1)} q + d \log^2 d, d, 0, 0, 0)$. Following that, we start building the tree register, and as in the proof of statement (2), in dependency of a given point in that process, we denote by $\mathcal{N}$ the initial segment of $\mathbb{N}_0$ consisting of all n for which $\mathfrak{D}_n$ is defined at that point. Because the associated rooted tree isomorphism types $\mathfrak{T}_n$ are pairwise distinct and are elements of the set $\{\mathrm{Expand}(\mathfrak{T}_{i,h}) : i \in \{0, 1, \ldots, d - 1\}, h \in \{0, 1, \ldots, H_i\}\}$, we have

$$|\mathcal{N}| \leq d \cdot (\max_i H_i + 1) \leq d^2 \,\mathrm{mpe}(q - 1) + d \in O(d^2 \,\mathrm{mpe}(q - 1))$$

(for the bound on H_i, see formula (3.2) in Section 3.3). In particular,

$$N \in O(d^2 \operatorname{mpe}(q-1))$$

in the end, as asserted.

To build the register, we go through the elements $(i,\ell) \in \bar{\mathcal{L}}$ with $i < d$ (we note that $\operatorname{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ is simply trivial and is not even recorded in the register by definition), and for each of them, we do the following. First, we compute the exact value of H_i, the maximum tree height above a periodic vertex in $\bigcup_{t=0}^{\ell-1} C_{i_t}$, where $(i_0, i_1, \ldots, i_{\ell-1})$ with $i = i_0$ is the $\bar{f}$-cycle of i. We note that by Theorem 3.2.1 and the paragraph before it (which was worked out in detail in Section 4.1), we have the following. For each $t \in \{0,1,\ldots,\ell-1\}$, the trees above periodic vertices in C_{i_t} are pairwise isomorphic and thus of a common height $\mathcal{H}_{i_t}$. We have $H_i = \max\{\mathcal{H}_{i_t} : t = 0,1,\ldots,\ell-1\}$, so we compute the numbers $\mathcal{H}_{i_t}$ in order to get H_i. At this point, we note that in fact, in Section 5.2.2, we already described how to find H_i through a binary search. However, here we are also interested in storing the procreation numbers $\operatorname{proc}_{i_t,k}$ for $t \in \{0,1,\ldots,\ell-1\}$ and $k \in \{1,2,\ldots,\mathcal{H}_{i_t}\}$, whence we do *not* use binary search to skip steps.

We remind the reader that for arbitrary $t \in \mathbb{Z}$, the notation i_t is shorthand for $i_{t \bmod \ell}$. For $t = 0,1,\ldots,\ell-1$ and successively for $k = 1,2,\ldots$, we compute (and store) the procreation number (see Theorem 3.2.1)

$$\operatorname{proc}_{i_t,k} = \frac{\gcd(\prod_{r=0}^{k-1} \alpha_{i_{t-k+r}}, s)}{\gcd(\prod_{r=0}^{k-2} \alpha_{i_{t-k+r}}, s)}$$

until $\operatorname{proc}_{i_t,k} = 1$ for the first time for a given t, which happens precisely when $k = \mathcal{H}_{i_t} + 1$. If we store the values of the two products appearing in the formula for $\operatorname{proc}_{i_t,k}$, then the computation of each $\operatorname{proc}_{i_t,k}$ only involves $O(1)$ multiplications and thus has a bit operation cost in $O(\log^{1+o(1)} q)$ by Lemma 5.1.5 (3,8). Therefore, and because $\mathcal{H}_{i_t} \in O(\ell \operatorname{mpe}(q-1))$, we can compute each individual $\mathcal{H}_{i_t}$ using $O(\ell \operatorname{mpe}(q-1) \log^{1+o(1)} q)$ bit operations, whence the computation of H_i in total takes $O(\ell^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q)$ bit operations.

Once H_i has been computed, we start adding the information associated with the $\bar{f}$-cycle of i to our tree register. More precisely, we do the following successively for $h = 0,1,\ldots,H_i$. If $h = 0$, then $\operatorname{Expand}(\mathfrak{T}_{i_t,h})$ is trivial for all t, so in case $\mathcal{N} = \emptyset$ (which only happens for the first pair $(i,\ell) \in \bar{\mathcal{L}}$ we consider), we set $\mathfrak{D}_0 := \emptyset$, causing $\mathfrak{T}_0$ to be the trivial rooted tree (in particular, $\operatorname{ht}(\mathfrak{T}_0) = 0$), and we initialize some variables as follows.

- We set

$$S_{0,\text{trans}} := \begin{cases} \{i_0, i_1, \ldots, i_{\ell-1}\}, & \text{if } H_i > 0, \\ \emptyset, & \text{otherwise.} \end{cases}$$

- We set $S_{0,\mathrm{per}} := \{i_t : t \in \{0, 1, \ldots, \ell - 1\}, \mathcal{H}_{i_t} = 0\}$.

We remind the reader that we want the arrays representing $S_{0,\mathrm{trans}}$ and $S_{0,\mathrm{per}}$ to be sorted, so one should apply the sorting algorithm from Lemma 5.1.5 (10), which takes $O(d \log^2 d)$ bit operations for each array.

In the other case, where $\mathcal{N} \neq \emptyset$, we do the following.

- If $H_i > 0$, we add $i_0, i_1, \ldots, i_{\ell-1}$ to the already defined set $S_{0,\mathrm{trans}}$ as new elements (technically speaking, we sort $\{i_0, i_1, \ldots, i_{\ell-1}\}$ and merge it with $S_{0,\mathrm{trans}}$).
- We also add all indices i_t, for $t \in \{0, 1, \ldots, \ell - 1\}$, such that $\mathcal{H}_{i_t} = 0$ to the already defined set $S_{0,\mathrm{per}}$ as new elements.

Using Lemma 5.1.5 (10,11), one sees that dealing with the case $h = 0$ as a whole only takes $O(\ell \log^2 d + d \log d)$ bit operations (for copying information and sorting/merging, as well as simple look-ups of the values $\mathcal{H}_{i_t}$).

Now we assume that $h \geq 1$. The edge-weighted rooted tree (isomorphism type) $\mathfrak{T}_{i_t,h}$ is drawn at the end of Section 4.1 (we draw the reader's attention to the case distinction between $h < H_i$ and $h = H_i$), and we compute the description

$$\mathfrak{D} = \mathfrak{D}(i_t, h)$$

of $\mathrm{Expand}(\mathfrak{T}_{i_t,h})$ as follows. First, we set

$$h' := \begin{cases} h, & \text{if } h < H_i, \\ \mathcal{H}_{i_t}, & \text{if } h = H_i, \end{cases}$$

which is the height of $\mathrm{Expand}(\mathfrak{T}_{i_t,h})$. It is also the number of edges in $\mathfrak{T}_{i_t,h}$ that have the root as their terminal vertex (i.e., the unweighted in-degree of that root). We do note that some of these edges may have weight 0. For $k = 0, 1, \ldots, h' - 1$, we compute

$$\mathrm{w}_k := \begin{cases} \mathrm{w}_{i_t,k} = \mathrm{proc}_{i_t,k+1} - \mathrm{proc}_{i_t,k+2}, & \text{if } h = H_i, \text{ or } h < H_i \text{ and } k < h' - 1, \\ \mathrm{proc}_{i_t,h}, & \text{if } h < H_i \text{ and } k = h' - 1, \end{cases}$$

which is the weight of the $(k + 1)$-th edge in $\mathfrak{T}_{i_t,h}$ (counted from the left in the drawing) that has the root as its terminal vertex. These computations only require

$$O(\ell \cdot h' \cdot \log q) \subseteq O(\ell^2 \,\mathrm{mpe}(q - 1) \log q)$$

bit operations for all t together. We may then set

$$\mathfrak{D} := \{(\bar{n}_{i_t,k}, \mathrm{w}_k) : k = 0, 1, \ldots, h' - 1\},$$

where $\bar{n}_{i_t,k}$ is the unique $n \in \mathcal{N}$ such that

$$\mathfrak{T}_n = \mathrm{Expand}(\mathfrak{T}_{i_t,k}),$$

i.e., such that $i_t \in S_{n,\mathrm{trans}}$ and $\mathrm{ht}(\mathfrak{T}_n) = k$. Assuming that the heights of the various $\mathfrak{T}_n$ are stored whenever the register is updated, the computation of $\mathfrak{D}$ takes

$$\begin{aligned} &O(h' \cdot (|\mathcal{N}| \cdot \log^2 d + \log\log q)) \\ &\quad \subseteq O(\ell \operatorname{mpe}(q-1) \cdot (d^2 \operatorname{mpe}(q-1) \cdot \log^2 d + \log\log q)) \\ &\quad \subseteq O(\ell d^2 \log^2 d \operatorname{mpe}(q-1)^2 + \ell \operatorname{mpe}(q-1) \log\log q) \end{aligned}$$

bit operations for a single t (needed for determining the $\bar{n}_{i_t,k}$), hence

$$O(\ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3 + \ell^2 d \operatorname{mpe}(q-1)^2 \log\log q)$$

bit operations for all t and h together.

Once $\mathfrak{D}$ has been computed, we need to check whether it already occurs among the $\mathfrak{D}_n$ for $n \in \mathcal{N}$ (and update the register accordingly). If we sort $\mathfrak{D}$ lexicographically, we may compare it with a given $\mathfrak{D}_n$ through linear comparison of entries, and so checking whether $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$ takes

$$\begin{aligned} &O(h' \log h' \log q + |\mathcal{N}| h' \log q) \subseteq O(d^2 \operatorname{mpe}(q-1) \cdot \ell \operatorname{mpe}(q-1) \cdot \log q) \\ &\quad = O(\ell d^2 \operatorname{mpe}(q-1)^2 \log q) \end{aligned}$$

bit operations for a single t, hence

$$\begin{aligned} O(\ell H_i \cdot \ell d^2 \operatorname{mpe}(q-1)^2 \log q) &\subseteq O(\ell^2 d^3 \operatorname{mpe}(q-1)^3 \log q) \\ &\subseteq O(\ell d^4 \operatorname{mpe}(q-1)^3 \log q) \end{aligned}$$

bit operations for all t and h together. This last O-expression dominates every other bit operation cost given in this complexity analysis except the cost

$$O(\ell^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q) \subseteq O(\ell d \operatorname{mpe}(q-1) \log^{1+o(1)} q)$$

of computing H_i, and the total cost $O(\ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3)$ of computing the descriptions $\mathfrak{D}$. Therefore, the bit operation cost of these computations is in

$$O(\ell d \operatorname{mpe}(q-1) \log^{1+o(1)} q + \ell d^4 \operatorname{mpe}(q-1)^3 \log q + \ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3)$$

for a given (i, ℓ) and h. Using that $\sum_{(i,\ell)\in\bar{\mathfrak{L}}} \ell = d$, the total bit operation cost of computing the type-II tree register for f is in

$$\begin{aligned} &O(d \log^{1+o(1)} q + d^2 \log^2 d \\ &\quad + \sum_{(i,\ell)\in\bar{\mathfrak{L}}} (\ell d \operatorname{mpe}(q-1) \log^{1+o(1)} q + \ell d^4 \operatorname{mpe}(q-1)^3 \log q \\ &\qquad + \ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3)) \\ &\quad \subseteq O(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^5 \operatorname{mpe}(q-1)^3 \log q + d^5 \log^2 d \operatorname{mpe}(q-1)^3), \end{aligned}$$

as asserted. The "In particular" statement holds because $\operatorname{mpe}(q-1) \in O(\log q)$. ∎

So far, everything has been of a q-bounded query complexity that is polynomial in $\log q$ and d. The quantity $\tau(q-1)$, which is generally superpolynomial in $\log q$, enters through the following auxiliary result.

Proposition 5.3.2.3. *Let m be a positive integer, and let $A : x \mapsto ax + b$ be an affine map of $\mathbb{Z}/m\mathbb{Z}$. The cycle type of $A_{|\mathrm{per}(A)}$, denoted by $\mathrm{CT}(A_{|\mathrm{per}(A)})$, can be computed within m-bounded query complexity*

$$(\tau(m)\log^{2+o(1)} m + \tau(m)^2 \log m, 0, \log m, 1, 0).$$

Proof. We start with two suitable m-bounded mord queries, one suitable m-bounded mdl query and a computation of the remainder of a upon division by 4 (which merely consists of scanning the two least significant digits of a). Together, these allow us to factor m, compute $\mathrm{ord}_{p^{\nu_p(m)}}(a)$ for all primes $p \mid m$ such that $p \nmid a$, factor $a-1$ and (only if $\nu_2(m) \geq 2$) compute the (unique) exponents $\varepsilon \in \{0,1\}$ and

$$e \in \{0, 1, \ldots, 2^{\nu_2(m)-2}\}$$

such that $a \equiv (-1)^\varepsilon 5^e \pmod{2^{\nu_2(m)}}$ (first, one determines ε simply by looking at the value of a modulo 4, and then e can be determined with an mdl query). The m-bounded query complexity of these computations is $(\log m, 0, 1, 1, 0)$.

Letting $m'' := \prod_{p|\gcd(a,m)} p^{\nu_p(m)}$ and $m' := m/m''$ (which we do not need to actually compute), we observe the following. Because $A \bmod m''$ has a unique periodic point (see Lemma 2.1.14), we find that $\mathrm{CT}(A_{|\mathrm{per}(A)}) = \mathrm{CT}(A \bmod m')$, and so we compute the latter. This allows us to assume without loss of generality that A is a permutation of $\mathbb{Z}/m\mathbb{Z}$. For each prime $p \mid m$, we set $A_{(p)} := A \bmod p^{\nu_p(m)}$. Since A is given via its coefficients a and b, computing $A_{(p)}$ takes a mere $O(\log^{1+o(1)} m)$ bit operations per p for obtaining the remainders of a and b upon division by $p^{\nu_p(m)}$. Hence, computing all reductions $A_{(p)}$ takes $O(\log^{2+o(1)} m)$ bit operations.

Formulas for $\mathrm{CT}(A_{(p)})$ were given in [15, Tables 3 and 4], and since we know $\mathrm{ord}_{p^{\nu_p(m)}}(a)$ for all $p \mid m$ from our initial mord query, these formulas allow us to compute $\mathrm{CT}(A_{(p)})$ for a given p within m-bounded query complexity

$$(\log^{2+o(1)}(p^{\nu_p(m)}) + \nu_p(m)\log^{1+o(1)}(p^{\nu_p(m)}), 0, 1, 0, 0).$$

Indeed, taking a closer look at those formulas, we see that apart from the information computed in the first paragraph of this proof, we only need to compute a single power and carry out some simpler arithmetic (taking $p^{\nu_p(m)}$-bounded query complexity $(\log^{2+o(1)}(p^{\nu_p(m)}), 0, 1, 0, 0)$), followed by $O(\nu_p(m))$ iterations of a loop, each consisting of $O(1)$ basic arithmetic operations taking $O(\log^{1+o(1)}(p^{\nu_p(m)}))$ bit operations each. For all p together, computing $\mathrm{CT}(A_{(p)})$ has m-bounded query complexity $(\log^{2+o(1)} m, 0, \log m, 0, 0)$. We also note that each $\mathrm{CT}(A_{(p)})$ is a monomial with at most $\nu_p(m)+1$ factors, and that our computation process allows us to store $\mathrm{CT}(A_{(p)})$ with all cycle lengths fully factored.

Now, we may compute $\mathrm{CT}(A)$ via the formula $\mathrm{CT}(A) = \divideontimes_{p|m} \mathrm{CT}(A_{(p)})$, where $\divideontimes$ denotes the Wei–Xu product from [86, Definition 2.2 on pp. 182f.]. This can be done by looping over the $O(\prod_{p|m}(v_p(m)+1)) \subseteq O(\tau(m))$ tuples formed by choosing one variable power in the factorization of each $\mathrm{CT}(A_{(p)})$ and computing the Wei–Xu product of those variable powers (which is itself a variable power) according to [86, formula (2.9) in Lemma 2.3 (b)]. Doing so requires us to compute the least common multiple of the involved cycle lengths, which takes $O(\log^{2+o(1)} m)$ bit operations because those cycle lengths are already fully factored (see also the paragraph on the computation of l in the proof of Corollary 5.3.1.2), followed by $O(\log m)$ integer multiplications and divisions for computing the exponent, which also take $O(\log^{2+o(1)} m)$ bit operations together. In total, the process of computing all relevant Wei–Xu products of variable powers takes $O(\tau(m)\log^{2+o(1)} m)$ bit operations. Once this is done, we need to compute the product of those variable powers, which means that $O(\tau(m))$ times, we need to multiply a monic monomial with $O(\tau(m))$ distinct variable power factors, each with index and exponent in $\{1, 2, \ldots, m\}$, with a single such variable power. Each such multiplication takes $O(\tau(m)\log m)$ bit operations, so the overall complexity of these computations, which result in $\mathrm{CT}(A)$, is in $O(\tau(m)^2 \log m)$. ∎

Remark 5.3.2.4. By our proof of Proposition 5.3.2.3, the cycle type of an affine map A of $\mathbb{Z}/m\mathbb{Z}$ is a product of at most $\tau(m)$ variable powers, and so A has at most $\tau(m)$ distinct cycle lengths.

As far as lower bounds on the maximum number of distinct cycle lengths of an affine map of $\mathbb{Z}/m\mathbb{Z}$ are concerned, let us fix a positive integer K and primes $2 < p_1 < p_2 < \cdots < p_K$ such that $\gcd(p_j, p_{j'} - 1) = 1$ for $1 \leq j < j' \leq K$ (such primes exist for each K by Dirichlet's theorem on primes in arithmetical progressions, see [8, Chapter 7]). For variable positive integers $v_1, v_2, \ldots, v_K$, we set $m := p_1^{v_1} \cdots p_K^{v_K}$ and consider an automorphism $A : x \mapsto ax$ of $\mathbb{Z}/m\mathbb{Z}$ such that a is a primitive root modulo $p_j^{v_j}$ for each j (it is possible to choose a like this because of the Chinese remainder theorem). By [15, Table 3], the cycle lengths of A mod $p_j^{v_j}$ are just the numbers of the form $(p_j - 1)p_j^{v_j'}$, where $v_j' \in \{0, 1, \ldots, v_j - 1\}$. Therefore, the cycle lengths of A are just the numbers of the form $\prod_{j=1}^{K}(p_j - 1) \cdot \prod_{j=1}^{K} p_j^{v_j'}$, whence A has $v_1 \cdots v_K$ distinct cycle lengths. We observe that this cycle length count is asymptotically equivalent to

$$(v_1 + 1) \cdots (v_K + 1) = \tau(m)$$

if $\min\{v_1, \ldots, v_K\} \to \infty$. Moreover, we note that

$$\log m = v_1 \log p_1 + \cdots + v_K \log p_K \leq (v_1 + \cdots + v_K) \log p_K.$$

Now, let us assume that $v_1 = v_2 = \cdots = v_K =: v \to \infty$. Then the number of distinct

cycle lengths of A is

$$v^K = \left(\frac{\log p_K \cdot K \cdot v}{\log p_K \cdot K}\right)^K \geq \left(\frac{\log m}{\log p_K \cdot K}\right)^K$$
$$= \frac{\log^K m}{(\log p_K \cdot K)^K} = c(K, p_K) \cdot \log^K m,$$

where $c(K, p_K) := (K \log p_K)^{-K}$. Because we can construct such a class of examples for each K, the maximum number of distinct cycle lengths of an affine map of $\mathbb{Z}/m\mathbb{Z}$ is in general not bounded from above by a polynomial in $\log m$.

Before we proceed further, we need another auxiliary concept and result.

Definition 5.3.2.5. Let $\vec{\mathfrak{x}} = (\mathfrak{x}_1, \mathfrak{x}_2, \ldots, \mathfrak{x}_n)$ be a finite sequence. A *period length of* $\vec{\mathfrak{x}}$ is a positive divisor m of n such that $\vec{\mathfrak{x}} = \diamond_{t=1}^{n/m}(\mathfrak{x}_1, \mathfrak{x}_2, \ldots, \mathfrak{x}_m)$, where $\diamond$ denotes concatenation (as in Section 3.3). The smallest positive integer that is a period length of $\vec{\mathfrak{x}}$ is denoted by $\mathrm{minperl}(\vec{\mathfrak{x}})$.

Remark 5.3.2.6. We note the following concerning Definition 5.3.2.5.

(1) The number $\mathrm{minperl}(\vec{\mathfrak{x}})$ is well defined because at the very least, the length n of $\vec{\mathfrak{x}}$ is a period length of it.

(2) All elements in $[\vec{\mathfrak{x}}]$, the cyclic equivalence class of the sequence $\vec{\mathfrak{x}}$ (see the second paragraph after Definition 1.4) have the same period lengths, in particular the same minperl-value, as $\vec{\mathfrak{x}}$. We denote this common minperl-value by $\mathrm{minperl}([\vec{\mathfrak{x}}])$.

We can bound the complexity of computing $\mathrm{minperl}(\vec{\mathfrak{x}})$ as follows.

Lemma 5.3.2.7. *Let $\vec{\mathfrak{x}} \in \{0, 1, \ldots, N-1\}^n$, given as a length n list of non-negative integers of bit length l_{bit}. Then* $\mathrm{minperl}(\vec{\mathfrak{x}})$ *can be computed using*

$$O(n \log n \log\log n(\log n + l_{\mathrm{bit}})) \subseteq O(n^{1+o(1)} l_{\mathrm{bit}})$$

bit operations.

Proof. We start by determining n, the number of entries of the tuple $\vec{\mathfrak{x}}$, in its binary representation with $\lfloor \log_2 n \rfloor + 1$ bits. The corresponding incrementation process involves altering

- only 1 bit in case the number to increment is even, i.e., in $O(\frac{n}{2})$ of the cases;
- only 2 bits in case the number to increment is congruent to 1 modulo 4, i.e., in $O(\frac{n}{4})$ of the cases;

and so on. Hence, the number of necessary bit operations to compute n is in

$$O\Big(\sum_{k=1} \infty \frac{k\,n}{2^k}\Big) = O(n).$$

Following that, we factor n deterministically. The current record for the bit operation cost of this is $O(n^{1/5+o(1)})$ due to Harvey and Hittmeir [29] (we could also use an mdl query for this factorization, but then the algorithm is not entirely classical, and $n^{1/5+o(1)}$ is majorized by other terms in this analysis anyway). The rest of the algorithm is analogous to the deterministic procedure for computing a modular multiplicative order described in the proof of Lemma 5.1.6 (2). More specifically, if $n = p_1^{v_1} \cdots p_K^{v_K}$ is the obtained factorization of n, then for $j = 1, 2, \ldots, K$, we perform a binary search to find the smallest $v_j' \in \mathbb{N}_0$ such that $p_j^{v_j'} \prod_{k \neq j} p_k^{v_k}$ is a period length of $\vec{\mathfrak{x}}$, which implies that $v_j' = v_{p_j}(\mathrm{minperl}(\vec{\mathfrak{x}}))$. For each given j, it takes $O(\log v_j) \subseteq O(\log \log n)$ iterations of the binary search loop until v_j' is found, and each iteration costs $O(n(\log n + l_{\mathrm{bit}}))$ bit operations. Because $K \in O(\log n)$, this means that the total bit operation cost of computing the numbers v_j' is in

$$O(n \log n \log \log n (\log n + l_{\mathrm{bit}})),$$

and $\mathrm{minperl}(\vec{\mathfrak{x}}) = \prod_{j=1}^{K} p_j^{v_j'}$ takes $O(\log n \cdot \log^{2+o(1)} n) = O(\log^{3+o(1)} n)$ bit operations to compute by Lemma 5.1.5 (3,6). ∎

We now give the precise definition of the compact description of the isomorphism type of Γ_f we aim to obtain.

Definition 5.3.2.8. Let f be a function $X \to X$, where X is some finite set, and let $\vec{\mathfrak{F}} = (\mathfrak{F}_n)_{n=0,1,\ldots,N}$ be a sequence of pairwise distinct finite rooted tree isomorphism types that contains all isomorphism types of the form $\mathrm{Tree}_{\Gamma_f}(x)$ for $x \in \mathrm{per}(f)$. The *tree necklace list for f relative to $\vec{\mathfrak{F}}$* is the unique set $\{([\vec{\mathfrak{n}}_k], l_k, \mathfrak{m}_k) : k = 1, 2, \ldots, N'\}$ of triples such that the following hold.

(1) $[\vec{\mathfrak{n}}_k] = [\mathfrak{n}_{k,1}, \mathfrak{n}_{k,2}, \ldots, \mathfrak{n}_{k,\ell_k'}]$ is a cyclic sequence of numbers in $\{0, 1, \ldots, N\}$ such that $\mathrm{minperl}([\vec{\mathfrak{n}}_k]) = \ell_k'$.

(2) l_k and $\mathfrak{m}_k$ are positive integers, and l_k is a multiple of ℓ_k'.

(3) In Γ_f, there are precisely $\mathfrak{m}_k$ connected components corresponding to the cyclic sequence of rooted tree isomorphism types

$$[\diamond_{t=1}^{l_k/\ell_k'}(\mathfrak{F}_{\mathfrak{n}_{k,1}}, \mathfrak{F}_{\mathfrak{n}_{k,2}}, \ldots, \mathfrak{F}_{\mathfrak{n}_{k,\ell_k'}})].$$

(4) For each connected component of Γ_f, there is a $k \in \{1, 2, \ldots, N'\}$ such that the said connected component corresponds to

$$[\diamond_{t=1}^{l_k/\ell_k'}(\mathfrak{F}_{\mathfrak{n}_{k,1}}, \mathfrak{F}_{\mathfrak{n}_{k,2}}, \ldots, \mathfrak{F}_{\mathfrak{n}_{k,\ell_k'}})].$$

If f is an index d generalized cyclotomic mapping of the finite field $\mathbb{F}_q$ such that f is of special type I or II, respectively, and if $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ is a type-I or -II tree register for f, then associated with $\mathfrak{R}$, we have the sequence $(\mathfrak{F}_n)_{n=0,1,\ldots,N}$

of rooted tree isomorphism types where $\mathfrak{I}_n$ has the compact description $\mathfrak{D}_n$. In that case, the tree necklace list for f relative to $\vec{\mathfrak{I}}$ is also called one *relative to* $\mathfrak{R}$.

Remark 5.3.2.9. We make the following comments concerning Definition 5.3.2.8.

(1) The uniqueness of the tree necklace list for f relative to $\vec{\mathfrak{I}}$ is not hard to prove, but it does require that $\mathrm{minperl}([\vec{\mathfrak{n}}_k]) = \ell'_k$ for all k. For example, without this property, for any $\vec{\mathfrak{I}}$ of length $N+1 \geq 2$, both $\{([0,1],4,1)\}$ and $\{([0,1,0,1],4,1)\}$ would be tree necklace lists relative to $\vec{\mathfrak{I}}$ for a suitable function f.

(2) For $j = 1, 2$, let X_j be a finite set and f_j a function $X_j \to X_j$. Moreover, let $\vec{\mathfrak{I}}$ be a finite sequence of pairwise distinct, finite rooted tree isomorphism types such that for $j = 1, 2$, each $\mathrm{Tree}_{\Gamma_{f_j}}(x)$ for $x \in \mathrm{per}(f_j)$ occurs in $\vec{\mathfrak{I}}$. For $j = 1, 2$, let $\mathfrak{N}_j$ be the unique tree necklace list for f_j relative to $\vec{\mathfrak{I}}$. It is not hard to prove that $\mathfrak{N}_1 = \mathfrak{N}_2$ if and only if $\Gamma_{f_1} \cong \Gamma_{f_2}$.

(3) Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$ that is of special type I or II. We need to fix suitable bit string encodings of the elements of a tree necklace list for f, making the following conventions. By assumption, if $i \in \{0, 1, \ldots, d\}$ has cycle length ℓ under f, then the cyclic sequence of rooted tree isomorphism types encoding the connected component of Γ_f containing any f-periodic vertex from C_i has minimal period some divisor of ℓ. In particular, the said minimal period is always at most d. Moreover, we assume that $\vec{\mathfrak{I}}$ stems from a recursive tree description list $\vec{\mathfrak{D}}$ that is part of a type-I or -II tree register $\mathfrak{R}$ for f. In $\vec{\mathfrak{D}}$, there is a common bit length to represent numbers from $\{0, 1, \ldots, N\}$ (see the remarks after Definition 5.3.2.1); we denote that bit length by l_{bit} and observe that

$$l_{\mathrm{bit}} = \begin{cases} \lfloor \log_2 d \rfloor + 1 \in O(\log d), & \text{if } \mathfrak{R} \text{ has type I,} \\ \lfloor \log_2(d^2 \lfloor \log_2 q \rfloor + d) \rfloor + 1 \in O(\log d + \log\log q), & \text{if } \mathfrak{R} \text{ has type II.} \end{cases}$$

A cyclic sequence $[\vec{\mathfrak{n}}] = [\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{\ell'}]$ as above is the first entry of an element of the tree necklace list for f relative to $\mathfrak{R}$; we assume that the associated ordered sequence $\vec{\mathfrak{n}} = (\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{\ell'})$ is lexicographically minimal among all ordered sequences in its cyclic equivalence class $[\vec{\mathfrak{n}}]$. We encode $[\vec{\mathfrak{n}}]$ as follows. We take the ordered sequence $\vec{\mathfrak{n}}$ and fill it up with entries -1 (a dummy value) until it has d entries. We then print a bit string that is a concatenation of encodings of the entries of this extended sequence. We use $l_{\mathrm{bit}} + 1$ bits to denote each entry, where an entry other than -1 is represented by an ancillary bit 1, followed by the length l_{bit} binary digit representation of that entry from $\mathfrak{R}$. On the other hand, an entry -1 is represented by a string of $l_{\mathrm{bit}} + 1$ zeroes. For example, if $l_{\mathrm{bit}} = 3$ and $d = 5$, then the bit string encoding

of $[6, 3, 4] = [3, 4, 6]$ in the corresponding tree necklace list is

$$10111100111000000000.$$

On the other hand, the second and third entries of elements of any tree necklace list for f are positive integers that are at most q, and we simply use their standard binary representations with $\lfloor \log_2 q \rfloor + 1$ digits to represent them; these may be directly concatenated with the bit string encoding of $[\vec{\mathfrak{n}}]$. With these conventions, all elements of a given tree necklace list for f are bit strings of the same bit length, namely

$$d(l_{\text{bit}} + 1) + 2(\lfloor \log_2 q \rfloor + 1) \in \begin{cases} O(d \log d + \log q), & \text{if } \mathfrak{R} \text{ has type I,} \\ O(d \log d + d \log\log q + \log q), & \text{if } \mathfrak{R} \text{ has type II,} \end{cases}$$

which allows us to use the sorting algorithm from Lemma 5.1.5 (10) to sort them lexicographically. Moreover, the lexicographic ordering of those bit strings corresponds to the "natural" lexicographic ordering of the elements of the associated (abstract) tree necklace list (using the lexicographic ordering among lexicographically minimal representatives of cyclic sequences in the first component, and the usual linear ordering of integers in the second and third component). It should be noted that our approach involves some padding, and this could be avoided through using [6, Algorithm 3.2 on pp. 80f.], which is a more general lexicographic sorting algorithm that does not require the bit strings from the input to be of a common length. However, in terms of the O-class of the complexity bounds we derive, it does not make a difference.

Remark 5.3.2.9 (2) guarantees that tree necklace lists are *injective* descriptions of digraph isomorphism types of functional graphs, but they may not always be compact. Indeed, they contain as many elements as there are distinct isomorphism types of connected components of the said functional graph, and in the case of the functional graph Γ_f of a generalized cyclotomic mapping f of $\mathbb{F}_q$ of a fixed index d, the maximum number of such connected components is at least the maximum number of distinct cycle lengths which an affine permutation of $\mathbb{Z}/s\mathbb{Z}$ can achieve. That latter number can, a priori, be superpolynomial in $s = (q-1)/d$ (and thus in q if d is fixed), see Remark 5.3.2.4. We note however, that the moduli considered in Remark 5.3.2.4 are of a special form, and it is not clear whether the construction from Remark 5.3.2.4 can still be carried out if, additionally, all constructed moduli must be of the form $(q-1)/d$ for some prime power q with $d \mid q-1$, where $d \in \mathbb{N}^+$ is fixed. Moreover, by our Proposition 5.3.1.4, as long as one is willing to exclude a small positive asymptotic fraction of prime powers, then $\tau(q-1)$, which is an upper bound on the

number of distinct cycle lengths of an affine map of $\mathbb{Z}/s\mathbb{Z}$ (see the proof of Proposition 5.3.2.3), *is* polynomial in $\log q$. Even for such prime powers q, the number of distinct isomorphism types of connected components of Γ_f itself could be superpolynomial in $\log q$, however. We leave the problem of finding a concrete infinite class of examples that confirms this open; see also Problems 6.3.3 and 6.3.4.

Nonetheless, if f is of special type I or II, then the following key result implies that one may compute a tree register $\mathfrak{R}$ for f and, subsequently, compute and print the tree necklace list for f relative to $\mathfrak{R}$ within a q-bounded query complexity that is polynomial in $\log q$, d and $\tau(q-1)$.

Theorem 5.3.2.10. *Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$.*

(1) *We assume that f is of special type* I. *Then one can compute within a q-bounded query complexity of*

$$(d^3\log^2 d + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

or a q-bounded Las Vegas dual complexity of

$$(d^3\log^2 d + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q),$$

*a type-*I *tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d)$, as well as the tree necklace list of f relative to $\mathfrak{R}$.*

(2) *We assume that f is of special type* II. *Then one can compute within a q-bounded query complexity of*

$$(d^5\log^2 d\,\mathrm{mpe}(q-1)^3 + (d^5\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

or a q-bounded Las Vegas dual complexity of

$$(d^5\log^2 d\,\mathrm{mpe}(q-1)^3 + (d^5\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q),$$

*a type-*II *tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ of f with*

$$N \in O(d^2\,\mathrm{mpe}(q-1)),$$

as well as the tree necklace list of f relative to $\mathfrak{R}$.

Proof. We prove both statements simultaneously, referring with "case I", respectively, "case II", to the situation described in statement (1), respectively, (2). First, we compute $\bar{f}$, the affine maps A_i, and a tree register $\mathfrak{R}$ for f of the desired type and with the asserted bound on N, taking q-bounded query complexity

- $(d^3 \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0)$ in case I,
- $(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^5 \operatorname{mpe}(q-1)^3 \log q + d^5 \log^2 d \operatorname{mpe}(q-1)^3, d, 0, 0, 0)$ in case II

by Proposition 5.1.8 and Lemma 5.3.2.2 (2,4). Then we compute a CRL-list $\bar{\mathfrak{L}}$ of $\bar{f}$ together with the cycles of $\bar{f}$, taking $O(d \log^2 d)$ bit operations by the argument at the beginning of Section 5.2.1. For each $(i, \ell) \in \bar{\mathfrak{L}}$, letting $(i_0, i_1, \dots, i_{\ell-1})$ with $i = i_0$ denote the $\bar{f}$-cycle of i determined earlier, we compute the tree necklace list $\mathfrak{N}_i$, relative to $\mathfrak{R}$, for the restriction of f to $\bigcup_{t=0}^{\ell-1} C_{i_t}$ as follows.

If $i = d$, we simply set $\mathfrak{N}_i := \{([\mathfrak{n}], 1, 1)\}$, where $\mathfrak{n} \in \{0, 1, \dots, N\}$ is the positive integer that represents $\operatorname{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ in $\mathfrak{R}$. We observe that in case II, where $\operatorname{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ is trivial, one has $\mathfrak{n} = 0$ necessarily, whereas in case I, the number $\mathfrak{n}$ is uniquely characterized by the inclusion $d \in S_{\mathfrak{n}}$, and can thus be determined with $O(d \log d)$ bit operations (using that each $\mathfrak{n}$ is represented by a bit string of length in $O(\log d)$).

Now we assume that $i < d$. Then we compute $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$, taking $O(d \log^{1+o(1)} q)$ bit operations (see Section 5.2.1, page 118), as well as the cycle type $\operatorname{CT}((\mathcal{A}_i)_{|\operatorname{per}(\mathcal{A}_i)}) = x_1^{e_{i,1}} x_2^{e_{i,2}} \cdots x_s^{e_{i,s}}$ (where some of the $e_{i,j}$ may be 0), taking q-bounded query complexity

$$(\tau(q-1) \log^{2+o(1)} q + \tau(q-1)^2 \log q, 0, \log q, 1, 0)$$

by Proposition 5.3.2.3. Moreover, we compute the number sequence

$$\vec{\mathfrak{n}}_i = (\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \dots, \mathfrak{n}_{i_{\ell-1}}),$$

where $\mathfrak{n}_{i_t}$ is uniquely characterized by the inclusion

$$\begin{cases} i_t \in S_{\mathfrak{n}_{i_t}}, & \text{in case I,} \\ i_t \in S_{\mathfrak{n}_{i_t}, \operatorname{per}}, & \text{in case II,} \end{cases}$$

i.e., $\mathfrak{n}_{i_t}$ is the positive integer that represents, in the register $\mathfrak{R}$, the unique rooted tree isomorphism type above periodic vertices of Γ_f that are contained in C_{i_t}. The number of bit operations it takes to determine (and set the value of) $\mathfrak{n}_{i_t}$ for all $t = 0, 1, \dots, \ell - 1$ can be bounded as follows:

- in case I, it is in

$$O\left(d \cdot \sum_{n=0}^{N} |S_n| \cdot \log d + d\, l_{\text{bit}}\right) \subseteq O(d^2 \log d);$$

- in case II, it is in

$$O\left(d\left(N+\sum_{n=0}^{N}|S_{n,\mathrm{per}}|\right)\log d + d\,l_{\mathrm{bit}}\right) \subseteq O(d^3 \log d\,\mathrm{mpe}(q-1) + d\log\log q).$$

Next, we overwrite $\vec{\mathfrak{n}}_i$ with the unique lexicographically minimal number sequence in the same cyclic equivalence class. We can do by spelling the $O(\ell) \subseteq O(d)$ cyclic shifts of $\vec{\mathfrak{n}}_i$ out, then ordering them lexicographically as in Lemma 5.1.5 (10) and taking the first sequence in the sorted list. This process takes

$$\begin{aligned} O(d\log d \cdot d\,l_{\mathrm{bit}}) &= O(d^2 \log d\, l_{\mathrm{bit}}) \\ &\subseteq \begin{cases} O(d^2\log^2 d), & \text{in case I,} \\ O(d^2\log^2 d + d^2 \log d \log\log q), & \text{in case II} \end{cases} \end{aligned}$$

bit operations. Following that, we compute $\mathrm{minperl}(\vec{\mathfrak{n}}_i)$, which by Lemma 5.3.2.7 takes the following amount of bit operations:

$$\begin{aligned} &O(d\log d\log\log d\cdot(\log d + l_{\mathrm{bit}})) \\ &\subseteq \begin{cases} O(d\log^2 d\log\log d), & \text{in case I,} \\ O(d\log d\log\log d(\log d + \log\log q)), & \text{in case II.} \end{cases} \end{aligned}$$

We observe that the following is a valid choice for $\mathfrak{N}_i$:

$$\mathfrak{N}_i := \{([\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\mathrm{minperl}(\vec{\mathfrak{n}}_i)-1}}], \ell\cdot l', e_{i,l'}) : l' \in \{1,2,\ldots,s\}, e_{i,l'} > 0\}.$$

For further processing, rather than store $\mathfrak{N}_i$ literally as this list, it is more advantageous to store $\vec{\mathfrak{n}}'_i := [\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\mathrm{minperl}(\vec{\mathfrak{n}}_i)-1}}]$ and the list

$$\mathfrak{N}'_i := \{(\ell\cdot l', e_{i,l'}) : l' \in \{1,2,\ldots,s\}, e_{i,l'} > 0\}$$

separately, which takes

$$\begin{aligned} &O(\mathrm{minperl}(\vec{\mathfrak{n}}_i) l_{\mathrm{bit}} + \tau(q-1)\log^{1+o(1)} q) \\ &\subseteq \begin{cases} O(d\log d + \tau(q-1)\log^{1+o(1)} q), & \text{in case I,} \\ O(d(\log d + \log\log q) + \tau(q-1)\log^{1+o(1)} q), & \text{in case II} \end{cases} \end{aligned}$$

bit operations for carrying out the multiplications $\ell\cdot l'$ and copying data. This concludes our analysis of how to compute $\mathfrak{N}_i$, which in summary for each given i takes q-bounded query complexity

$$\begin{aligned} &(\tau(q-1)\log^{2+o(1)} q + (d+\tau(q-1))\log^{1+o(1)} q \\ &\quad + \tau(q-1)^2\log q + E(d,q), 0, \log q, 1, 0), \end{aligned}$$

where

$$E(d,q) := Nd\log d + d^2\log d\, l_{\text{bit}}$$
$$\in \begin{cases} O(d^2\log^2 d), & \text{in case I,} \\ O(d^3\log d\,\operatorname{mpe}(q-1) + d^2\log^2 d + d^2\log d\log\log q), & \text{in case II} \end{cases}$$

and the additional factor $\tau(q-1)$ in the bit operation cost is because there are only $\tau(s) \le \tau(q-1)$ distinct cycle lengths by the proof of Proposition 5.3.2.3 (see also the first few lines in Remark 5.3.2.4). For all i together, the q-bounded query complexity is the (component-wise) d-fold of that.

Finally, we need to compute the actual tree necklace list $\mathfrak{N}$ for f relative to $\mathfrak{R}$. We start by setting $\mathfrak{M} := \mathfrak{N} := \emptyset$ and $\vec{\mathfrak{n}}' := \{(\vec{\mathfrak{n}}'_i, i, \ell) : (i,\ell) \in \bar{\mathcal{L}}\}$ (where $\vec{\mathfrak{n}}'_d := [\mathfrak{n}]$, the first entry of the unique triple in $\mathfrak{N}_d$), followed by sorting $\vec{\mathfrak{n}}'$ lexicographically. Altogether, this takes $O(d^2\log d\, l_{\text{bit}})$ bit operations. Throughout the subsequently described process, $\mathfrak{N}$ is a (lexicographically sorted) initial segment of the tree necklace list that will be output, and $\mathfrak{M}$ is an initial segment of the list obtained by deleting repeated entries in the multiset $\{\vec{\mathfrak{n}}'_i : (i,\ell) \in \bar{\mathcal{L}}\}$. In particular, $\mathfrak{M}$ has $O(d)$ elements, each of which is a cyclic sequence of length in $O(d)$ each entry of which has bit length in $O(l_{\text{bit}})$ and which is given by its lexicographically minimal representative. We observe that two such cyclic sequences are equal if and only if their representatives are equal, so it takes $O(d\, l_{\text{bit}})$ bit operations to verify such an equality.

Now, we go through the triples $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$, and for each of them, we do the following. First, we check whether $\vec{\mathfrak{n}}'_i \in \mathfrak{M}$, which takes $O(d^2 l_{\text{bit}})$ bit operations. If so, we skip to the next triple in $\vec{\mathfrak{n}}'$, otherwise we proceed as follows. We add $\vec{\mathfrak{n}}'_i$ to $\mathfrak{M}$ as a new element (which takes $O(d\, l_{\text{bit}})$ bit operations for copying), and set $\mathfrak{V} := \mathfrak{N}'_i$, taking $O(\tau(q-1)\log q)$ bit operations.

Then, we go through the elements $(\vec{\mathfrak{n}}'_j, j, \ell') \in \vec{\mathfrak{n}}'$ that come after $(\vec{\mathfrak{n}}'_i, i, \ell)$, and for each of them, we do the following. We check whether $\vec{\mathfrak{n}}'_j = \vec{\mathfrak{n}}'_i$, which takes $O(d\, l_{\text{bit}})$ bit operations. If not, we skip to the next value of $(\vec{\mathfrak{n}}'_j, j, \ell')$, otherwise we proceed as follows. We go through the elements (l,k) of $\mathfrak{N}'_j$, and for each of them, we check whether l occurs as the first entry of some pair (l,k') of $\mathfrak{V}$; each such check takes $O(|\mathfrak{V}|\log q) \subseteq O(d\,\tau(q-1)\log q)$ bit operations. If so, we replace the unique element of $\mathfrak{V}$ of the form (l,k') by $(l,k+k')$; otherwise, we add (l,k) to $\mathfrak{V}$ as a new element.

Overall, the described loop over the elements of $\mathfrak{N}'_j$ takes

$$O(|\mathfrak{N}'_j| \cdot d\,\tau(q-1)\log q) \subseteq O(d\,\tau(q-1)^2\log q)$$

bit operations, and thus the loop over $(\vec{\mathfrak{n}}'_j, j, \ell')$ takes $O(d^2 l_{\text{bit}} + d^2\tau(q-1)^2\log q)$ bit operations. Once that loop is finished, we complete the loop over $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$ by adding, for each $(l,k) \in \mathfrak{V}$, the triple $([\vec{\mathfrak{n}}'_i], l, k)$ to $\mathfrak{N}$ as a new element – this copying

process takes

$$O(|\mathfrak{V}| \cdot (d\, l_{\text{bit}} + \log q)) \subseteq O(d\,\tau(q-1)(d\, l_{\text{bit}} + \log q))$$

bit operations.

At the end of the loop over $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$, the variable $\mathfrak{N}$ has its desired value, and the overall bit operation cost of this loop is in

$$O(d \cdot (d^2 l_{\text{bit}} + d^2\tau(q-1)l_{\text{bit}} + d^2\tau(q-1)^2 \log q)) = O(d^3\tau(q-1)^2 \log q). \blacksquare$$

Finally, we discuss the complexity of the digraph isomorphism problem. Let f_1 and f_2 be generalized cyclotomic mappings of $\mathbb{F}_q$, each of one of the special types I or II. We note that neither do f_1 and f_2 need to have the same index, nor are they necessarily both of the same special type. In order to decide whether $\Gamma_{f_1} \cong \Gamma_{f_2}$, we would like to compare a computed tree necklace list for f_1 with one for f_2. To that end, those tree necklace lists must be "synchronized", so that each n denotes the same rooted tree isomorphism type $\mathfrak{T}_n$ in each case. Here is a precise definition.

Definition 5.3.2.11. Let $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$ and $\vec{\mathfrak{D}}' = (\mathfrak{D}'_n)_{n=0,1,\ldots,N'}$ be recursive tree description lists, with associated rooted tree isomorphism type sequences

$$(\mathfrak{T}_n)_{n=0,1,\ldots,N} \text{ and } (\mathfrak{T}'_n)_{n=0,1,\ldots,N'}.$$

A *synchronization of* $\vec{\mathfrak{D}}$ *and* $\vec{\mathfrak{D}}'$ is a pair $(\vec{\mathfrak{D}}^+, \mathrm{i})$ such that the following hold.

(1) $\vec{\mathfrak{D}}^+ = (\mathfrak{D}^+_n)_{n=0,1,\ldots,N^+}$ is a recursive tree description list of which $\vec{\mathfrak{D}}$ is an initial segment (in particular, $N \leq N^+$). We denote by $(\mathfrak{T}^+_n)_{n=0,1,\ldots,N^+}$ the unique rooted tree isomorphism type sequence associated with $\vec{\mathfrak{D}}^+$.

(2) i is a function $\{0, 1, \ldots, N'\} \to \{0, 1, \ldots, N^+\}$ with

$$\{N+1, N+2, \ldots, N^+\} \subseteq \mathrm{im}(\mathrm{i})$$

such that for each $n \in \{0, 1, \ldots, N'\}$, one has $\mathfrak{T}'_n \cong \mathfrak{T}^+_{\mathrm{i}(n)}$.

In an implementation, we assume that each description $\mathfrak{D}_n$ or $\mathfrak{D}'_n$ is sorted by increasing first entries of its elements. We also assume that each of $\vec{\mathfrak{D}}$ and $\vec{\mathfrak{D}}'$ uses a common bit length, denoted by l_{bit} and l'_{bit}, respectively, for the binary representations of the numbers n. Because $N^+ \leq N + N'$, we use $\max\{l_{\text{bit}}, l'_{\text{bit}}\} + 1 \in O(l_{\text{bit}} + l'_{\text{bit}})$ bits for the numbers n in the synchronization $\vec{\mathfrak{D}}^+$. We note that the definition of a synchronization is asymmetric in the sense that a synchronization of $\vec{\mathfrak{D}}$ and $\vec{\mathfrak{D}}'$ is not necessarily also one of $\vec{\mathfrak{D}}'$ and $\vec{\mathfrak{D}}$. Complexity-wise, the following lemma shows that it is slightly more advantageous to have $N' \leq N$.

Lemma 5.3.2.12. *Let* $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$ *and* $\vec{\mathfrak{D}}' = (\mathfrak{D}'_n)_{n=0,1,\ldots,N'}$ *be recursive tree description lists such that for all* n,

- *each second entry of an element of $\mathfrak{D}_n$ or $\mathfrak{D}'_n$ is represented by a bit string of length at most m (a quantity that does not depend on n);*
- *each first entry of each element of $\mathfrak{D}_n$, respectively of $\mathfrak{D}'_n$, is represented by a bit string of length exactly l_{bit}, respectively, l'_{bit}, and*
- *within a given description $\mathfrak{D}_n$, respectively, $\mathfrak{D}'_n$, the second entries of elements of that description have a common bit length (so all pairs in $\mathfrak{D}_n$, respectively in $\mathfrak{D}'_n$, have the same bit length, which lies in $O(l_{\mathrm{bit}} + m)$, respectively in $O(l'_{\mathrm{bit}} + m)$).*

It takes $O((NN' \min\{N, N'\} + (\max\{N, N'\})^2 + (N')^2 l'_{\mathrm{bit}})(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m))$ bit operations to compute a synchronization of $\vec{\mathfrak{D}}$ and $\vec{\mathfrak{D}}'$.

Proof. Let $\mathfrak{I}_n$, respectively, $\mathfrak{I}'_n$, be the rooted tree isomorphism type described by $\mathfrak{D}_n$, respectively, by $\mathfrak{D}'_n$. In order to compute the synchronization, we proceed in a loop over $n = 0, 1, \ldots, N'$. At each given point in the process, the description $\mathfrak{D}^+_k$ (of the rooted tree isomorphism type $\mathfrak{I}^+_k$) is defined for all $k \in \mathcal{N}$, an initial segment of $\mathbb{N}_0$ that starts out as $\{0, 1, \ldots, N\}$, with $\mathfrak{D}^+_k := \mathfrak{D}_k$ for each $k \in \{0, 1, \ldots, N\}$, and will be $\{0, 1, \ldots, N^+\}$ in the end. We note that it takes

$$O(N^2(l_{\mathrm{bit}} + \log m)) \subseteq O((\max\{N, N'\})^2(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m))$$

bit operations (spent copying) to set $\mathcal{N}$ and the $\mathfrak{D}^+_k$ for $k \in \{0, 1, \ldots, N\}$ up. At any given point in the algorithm, the descriptions $\mathfrak{D}^+_k$ form a recursive tree description list denoted by $\vec{\mathfrak{D}}^+$ (which will have the desired value in the end). We also keep updating the value of $\mathrm{i} : \{0, 1, \ldots, n\} \to \mathcal{N}$, which starts out as the empty function $\emptyset$.

For $n = 0$, where $\mathfrak{D}'_n = \emptyset$ and its associated rooted tree is trivial, we simply set $\mathrm{i}(0) := 0$ without updating $\mathcal{N}$. Now let us assume that $n \geq 1$. Based on $\mathfrak{D}'_n$, we compute a new rooted tree description $\mathfrak{D}$ through replacing the first entry $k < n \leq N'$ of each given pair in $\mathfrak{D}'_n$ by $\mathrm{i}(k)$. Because $\mathfrak{D}'_n$ contains at most $n + 1 \in O(n)$ distinct pairs and we are handling non-negative integers of bit length in $O(l_{\mathrm{bit}} + l'_{\mathrm{bit}})$ here, computing $\mathfrak{D}$ as an unsorted list takes $O(n \cdot (l_{\mathrm{bit}} + l'_{\mathrm{bit}})) \subseteq O(N'(l_{\mathrm{bit}} + l'_{\mathrm{bit}}))$ bit operations overall (for a given n), and another $O(N' l'_{\mathrm{bit}}(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m))$ bit operations for sorting $\mathfrak{D}$.

Once $\mathfrak{D}$ has been computed in sorted form, we need to check whether the rooted tree $\mathfrak{I}'_n$ described by it with respect to $\vec{\mathfrak{D}}^+$ already occurs among the $\mathfrak{I}^+_k$. If $k > N$, then $\mathfrak{I}^+_k \cong \mathfrak{I}'_l$ for some $l \in \{0, 1, \ldots, n-1\}$, and thus $\mathfrak{I}^+_k \ncong \mathfrak{I}'_n$, as the isomorphism types $\mathfrak{I}'_t$ are pairwise distinct by assumption. Therefore, we only need to check the isomorphism $\mathfrak{I}^+_k \cong \mathfrak{I}'_n$ for $k \leq N$, where it is equivalent to $\mathfrak{I}_k \cong \mathfrak{I}'_n$ and further to $\mathfrak{D}_k = \mathfrak{D}$. For a given k, it takes $O(\min\{N, N'\}(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m))$ bit operations to check with a linear scan whether $\mathfrak{D}_k = \mathfrak{D}$ (using that both $\mathfrak{D}_k$ and $\mathfrak{D}$ are sorted), and so it can be checked with $O(N \min\{N, N'\}(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m))$ bit operations whether $\mathfrak{I}'_n \cong \mathfrak{I}^+_k$ for a (unique) $k \in \{0, 1, \ldots, N\}$. If so, we set $\mathrm{i}(n) := k$ without updating $\mathcal{N}$;

otherwise, we extend $\mathcal{N}$ by the new element $n' := \max \mathcal{N} + 1$ and set $\mathfrak{D}^+_{n'} := \mathfrak{D}$ and $\mathrm{i}(n) := n'$.

The overall bit operation cost of the described loop is in $O((NN' \min\{N, N'\} + (N')^2 l'_{\mathrm{bit}})(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m))$. We conclude the algorithm by outputting $(\vec{\mathfrak{D}}^+, \mathrm{i})$, where

$$\vec{\mathfrak{D}}^+ := (\mathfrak{D}^+_k)_{k \in \mathcal{N}}.$$

This takes

$$\begin{aligned} &O((N + N') \max\{N, N'\}(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m)) \\ &= O((\max\{N, N'\})^2 (l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m)) \end{aligned}$$

bit operations for copying. ∎

Corollary 5.3.2.13. *Let f_1 and f_2 be generalized cyclotomic mappings of a common finite field $\mathbb{F}_q$, say of index d_1 and d_2, respectively, and set $d := \max\{d_1, d_2\}$.*

(1) *If each f_j is of special type* I *or* II *(not necessarily both of the same type), then it takes q-bounded query complexity*

$$\begin{aligned} &((d^6 \operatorname{mpe}(q-1)^3 + d^3 \tau (q-1)^2) \log q + d^4 \operatorname{mpe}(q-1)^2 \log^{1+o(1)} q \\ &\quad + d\,\tau(q-1) \log^{2+o(1)} q, d, d \log q, d, 0), \end{aligned}$$

or q-bounded Las Vegas dual complexity

$$\begin{aligned} &((d^6 \operatorname{mpe}(q-1)^3 + d^3 \tau (q-1)^2) \log q + d^4 \operatorname{mpe}(q-1)^2 \log^{1+o(1)} q \\ &\quad + d\,\tau(q-1) \log^{2+o(1)} q + d \log^{8+o(1)} q, d \log^{4+o(1)} q, d \log^2 q), \end{aligned}$$

to decide whether $\Gamma_{f_1} \cong \Gamma_{f_2}$.

(2) *If both f_j are of special type I, then it takes q-bounded query complexity*

$$\begin{aligned} &(d^3 \log^2(d) + d^3 \tau (q-1)^2 \log q + d^2 \log^{1+o(1)} q \\ &\quad + d\,\tau(q-1) \log^{2+o(1)} q, d, d \log q, d, 0), \end{aligned}$$

or q-bounded Las Vegas dual complexity

$$\begin{aligned} &(d^3 \log^2(d) + d^3 \tau (q-1)^2 \log q + d^2 \log^{1+o(1)} q \\ &\quad + d\,\tau(q-1) \log^{2+o(1)} q + d \log^{8+o(1)} q, d \log^{4+o(1)} q, d \log^2 q), \end{aligned}$$

to decide whether $\Gamma_{f_1} \cong \Gamma_{f_2}$.

Proof. We denote the situation in statement (1) by "case (1)", and the one in statement (2) by "case (2)"; these must *not* be confused with cases I and II from the proof of Theorem 5.3.2.10. First, for $j = 1, 2$, we compute a suitable tree register $\mathfrak{R}_j$ of f_j

with $N_j + 1$ entries, together with an associated tree necklace list $\mathfrak{N}_j$ for f_j. By Theorem 5.3.2.10, this takes q-bounded query complexity

$$\begin{aligned}&(d^5 \log^2 d \operatorname{mpe}(q-1)^3 + (d^5 \operatorname{mpe}(q-1)^3 + d^3\tau(q-1)^2) \log q\\ &\quad + d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d\tau(q-1) \log^{2+o(1)} q,\\ &\quad d, d \log q, d, 0),\end{aligned}$$

in case (1), or

$$\begin{aligned}&(d^3 \log^2 d + d^3\tau(q-1)^2 \log q + d^2 \log^{1+o(1)} q + d\tau(q-1) \log^{2+o(1)} q,\\ &\quad d, d \log q, d, 0),\end{aligned}$$

in case (2).

Following that, we synchronize the underlying recursive tree description lists of the $\mathfrak{N}_j$. By Lemma 5.3.2.12, applied with $m := \log q$, and using the facts that $l_{\mathrm{bit}} \in O(\log q)$ and $l_{\mathrm{bit}}, l'_{\mathrm{bit}} \in O(\log d + \log\log q)$ in either case and that

$$\min\{N_1, N_2\} \leq \max\{N_1, N_2\} \in \begin{cases} O(d^2 \operatorname{mpe}(q-1)), & \text{in case (1),}\\ O(d), & \text{in case (2),}\end{cases}$$

we see that this can be done using

$$\begin{aligned}&O((d^6 \operatorname{mpe}(q-1)^3 + d^4 \operatorname{mpe}(q-1)^2 \log\log q)(\log d + \log\log q + \log q))\\ &\quad = O((d^6 \operatorname{mpe}(q-1)^3 + d^4 \operatorname{mpe}(q-1)^2 \log\log q) \log q)\end{aligned}$$

bit operations in case (1), or $O(d^3 \log q)$ bit operations in case (2). Following our convention on the bit lengths of indices n of descriptions $\mathfrak{D}_n^+$ in synchronizations, the computed synchronization uses a common bit length l_{bit}^+, which lies in $O(\log d + \log\log q)$ in case (1), and in $O(\log d)$ in case (2).

Next, based on $\mathfrak{N}_2$, we compute a modified tree necklace list $\mathfrak{N}'_2$ for f_2 with respect to the computed synchronization $(\vec{\mathfrak{D}}^+, \mathfrak{i})$ through replacing each number $\mathfrak{n}$ occurring as an entry in one of the cyclic sequences in $\mathfrak{N}_2$ by $\mathfrak{i}(\mathfrak{n})$, then replacing the underlying ordered sequence with the lexicographically minimal representative in the same cyclic equivalence class. Computing $\mathfrak{N}'_2$ in unsorted form takes

$$O(|\mathfrak{N}_2| \cdot (d l_{\mathrm{bit}}^+ + \log q) + d \cdot d \log d\, l_{\mathrm{bit}}^+) \subseteq O(d^2\tau(q-1) \log q + d^2 \log d \log q)$$

bit operations, and following that, we sort $\mathfrak{N}'_2$ lexicographically, which takes

$$O(d^3\tau(q-1) \log q)$$

bit operations through successively merging the $O(d)$ segments corresponding to a common, rewritten first entry (see Lemma 5.1.5 (11)).

Finally, we need to check whether $\mathfrak{N}_2' = \mathfrak{N}_1$, which only takes a linear scan thanks to $\mathfrak{N}_2'$ and $\mathfrak{N}_1$ both being lexicographically sorted (we note that the bit lengths of indices n in $\mathfrak{N}_1$ may not be the same as those in $\mathfrak{N}_2'$, but that is of course not a problem). The bit operation cost of this is in

$$O(\min\{|\mathfrak{N}_1|, |\mathfrak{N}_2'|\} \cdot (d\, l_{\text{bit}}^{+} + \log q)) \subseteq O(d\,\tau(q-1) \cdot d \log q) = O(d^2 \tau(q-1) \log q).$$

■

As we did throughout Section 5.2 and in Section 5.3.1, we conclude this subsection with pseudocode for all relevant algorithms introduced in it, specifying the query complexity (q-bounded or m-bounded, depending on the context) of each step. We start with the algorithm from Lemma 5.3.2.2 (1), which decides whether a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$ is of special type I.

1 Compute the affine maps $A_i : x \mapsto \alpha_i x + \beta_i$ of $\mathbb{Z}/s\mathbb{Z}$ associated with f.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 For each $i \in \{0, 1, \ldots, d-1\}$, do the following.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

2.1 If $a_i = 0_{\mathbb{F}_q}$, then skip to the next i.
QC: $(\log d, 0, 0, 0, 0)$.

2.2 If $\gcd(\alpha_i, s) > 1$, then output "false" and halt.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

3 Output "true" and halt.

Next, we list the steps of the algorithm from Lemma 5.3.2.2 (2), which computes, for a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$ that is of special type I, a type-I tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d)$.

1 Compute $s = (q-1)/d$, the induced function $\bar{f}$ on $\{0, 1, \ldots, d\}$ and the layers Layer_h, for $h \in \{0, 1, \ldots, \bar{H} - 1, \infty\}$, of $\bar{f}$ with respect to iteration.
QC: $(d \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Set $\mathcal{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

3 For each $i \in \{0, 1, \ldots, d\}$, determine its (ordered) list of $\bar{f}$-pre-images and its $\bar{f}$-transient pre-images.
QC: $(d \log d, 0, 0, 0, 0)$.

4 For each $h = 0, 1, \ldots, \bar{H} - 1, \infty$, do the following.
QC: $(d^3 \log^2 d + d \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1 If $h = 0$ or $\bar{H} = 0$, then do the following.

4.1.1 Add 0 to $\mathcal{N}$ as a new (the first) element, set $\mathfrak{D}_0 := \emptyset$ and $S_0 := \text{Layer}_h$.
QC: $(d \log d, 0, 0, 0, 0)$.

4.1.2 If $\bar{H} = 0$, then output $\mathfrak{R} := ((\mathfrak{D}_0, S_0))$ and halt.
QC: $(d \log d, 0, 0, 0, 0)$.

4.2 Else do the following.

4.2.1 For each $i \in \mathrm{Layer}_h$, letting $j_1, j_2, \ldots, j_K$ denote its $\bar{f}$-transient $\bar{f}$-pre-images, do the following.
QC: $(|\mathrm{Layer}_h|d^2 \log^2 d + \delta_{d \in \mathrm{Layer}_h} d \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.2.1.1 For each $t = 1, 2, \ldots, K$, do the following.
QC: $(d^2 \log^2 d, 0, 0, 0, 0)$.

4.2.1.1.1 Determine the unique number $\bar{n}_{j_t} \in \mathcal{N}$ such that $j_t \in S_{\bar{n}_{j_t}}$.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

4.2.1.2 If $i < d$, then do the following.

4.2.1.2.1 Set

$$\mathfrak{D} := \{(n, m) : n \in \{\bar{n}_{j_t} : 1 \le t \le K\}, \\ m = |\{t \in \{1, \ldots, K\} : \bar{n}_{j_t} = n\}| > 0\}.$$

QC: $(d \log^2 d, 0, 0, 0, 0)$.

4.2.1.3 Else do the following.

4.2.1.3.1 Set

$$\mathfrak{D} := \{(n, sm) : n \in \{\bar{n}_{j_t} : 1 \le t \le K\}, \\ m = |\{t \in \{1, \ldots, K\} : \bar{n}_{j_t} = n\}| > 0\}$$

QC: $(d \log^2 d + d \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.2.1.4 Check whether there is a (unique) $n \in \mathcal{N}$ such that $\mathfrak{D} = \mathfrak{D}_n$, and store this information (the truth value and n).
QC: $(d^2 \log d, 0, 0, 0, 0)$.

4.2.1.5 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

4.2.1.5.1 Merge the sorted lists $\{i\}$ and S_n.
QC: $(d \log d, 0, 0, 0, 0)$.

4.2.1.6 Else do the following.

(a) Set $n' := \max \mathcal{N} + 1$, and add n' to $\mathcal{N}$ as a new element.
QC: $(\log d, 0, 0, 0, 0)$.

(b) Set $\mathfrak{D}_{n'} := \mathfrak{D}$, and initialize $S_{n'} := \{i\}$.
QC: $(d \log d, 0, 0, 0, 0)$.

5 Output $\mathfrak{R} := ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,\max \mathcal{N}}$ and halt.
QC: $(d^2 \log d + d \log q, 0, 0, 0, 0)$.

Next, we give the (simple) pseudocode for the algorithm from Lemma 5.3.2.2 (3), which checks whether a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$ is of special type II.

1 Compute the induced function $\bar{f} : \{0, 1, \ldots, d\} \to \{0, 1, \ldots, d\}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Sort $\mathrm{im}(\bar{f})$, and check whether it is equal to $\{0, 1, \ldots, d\}$.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

The algorithm from Lemma 5.3.2.2 (4), for computing a type-II tree register for a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$ that is of special type II, has the following pseudocode.

1 Compute the induced function $\bar{f}$ and the affine maps $A_i : x \mapsto \alpha_i x + \beta_i$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Compute a CRL-list $\bar{\mathcal{L}}$ of $\bar{f}$ and, in the process, store the cycles of $\bar{f}$.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

3 Set $\mathcal{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

4 For each $(i, \ell) \in \bar{\mathcal{L}}$, do the following.
QC: $(d^5 \log^2 d \,\mathrm{mpe}(q-1)^3 + d^5 \,\mathrm{mpe}(q-1)^3 \log q + d^2 \,\mathrm{mpe}(q-1) \log^{1+o(1)} q$, $0, 0, 0, 0)$.

4.1 For each $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(\ell^2 \,\mathrm{mpe}(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.1 Set $\mathrm{prod}_{\mathrm{upper}} := \alpha_{i_{t-1}}$.
QC: $(\log q, 0, 0, 0, 0)$.

4.1.2 Set $\mathrm{prod}_{\mathrm{lower}} := 1$.
QC: $(1, 0, 0, 0, 0)$.

4.1.3 For each $k = 1, 2, \ldots$ (no explicit upper bound for this for-loop a priori, though by design it will stop at $k = \mathcal{H}_{i_t} + 1 \in O(\ell \,\mathrm{mpe}(q-1))$ – for the "big-O", see bound (3.2)), do the following.
QC: $(\ell \,\mathrm{mpe}(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.3.1 Compute

$$\mathrm{proc}_{i_t,k} := \frac{\gcd(\mathrm{prod}_{\mathrm{upper}}, s)}{\gcd(\mathrm{prod}_{\mathrm{lower}}, s)}.$$

QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.3.2 If $\mathrm{proc}_{i_t,k} = 1$, then do the following.

4.1.3.2.1 Set $\mathcal{H}_{i_t} := k - 1$.
QC: $(\log d + \log\log q, 0, 0, 0, 0)$.

4.1.3.2.2 Exit the loop for k, and skip to the next t.
QC: $(1, 0, 0, 0, 0)$.

4.1.3.3 Else do the following.

(a) Set $\mathrm{prod}_{\mathrm{lower}} := \mathrm{prod}_{\mathrm{upper}}$.
QC: $(\log q, 0, 0, 0, 0)$.

(b) Set $\mathrm{prod}_{\mathrm{upper}} := \mathrm{prod}_{\mathrm{upper}} \cdot \alpha_{i_{t-k-1}}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.2 Compute $H_i := \max\{\mathcal{H}_{i_t} : t = 0, 1, \ldots, \ell - 1\}$.
QC: $(\ell(\log d + \log\log q), 0, 0, 0, 0)$.

4.3 For $h = 0, 1, \ldots, H_i$, do the following.
QC: $(\ell^2 d^3 \log^2 d \,\mathrm{mpe}(q-1)^3 + \ell d^4 \,\mathrm{mpe}(q-1)^3 \log q, 0, 0, 0, 0)$.

4.3.1 If $h = 0$, then do the following.

4.3.1.1 If $\mathcal{N} = \emptyset$, then do the following.

4.3.1.1.1 Set $\mathfrak{D}_0 := \emptyset$ and $\mathrm{ht}_0 := 0$.
QC: $(1, 0, 0, 0, 0)$.

4.3.1.1.2 If $H_i > 0$, then set $S_{0,\mathrm{trans}} := \{i_0, i_1, \ldots, i_{\ell-1}\}$, sorted. Otherwise, set $S_{0,\mathrm{trans}} := \emptyset$.
QC: $(\ell \log^2 d, 0, 0, 0, 0)$.

4.3.1.1.3 Set $S_{0,\mathrm{per}} := \{i_t : \mathcal{H}_{i_t} = 0\}$, sorted.
QC: $(\ell \log^2 d, 0, 0, 0, 0)$.

4.3.1.2 Else do the following.

4.3.1.2.1 If $H_i > 0$, then sort $\{i_0, i_1, \ldots, i_{\ell-1}\}$ and merge it with $S_{0,\mathrm{trans}}$.
QC: $(\ell \log^2 d + d \log d, 0, 0, 0, 0)$.

4.3.1.2.2 Create a list of all indices i_t, for $t = 0, 1, \ldots, \ell - 1$, such that $\mathcal{H}_{i_t} = 0$, then sort it and merge it with $S_{0,\mathrm{per}}$.
QC: $(\ell \log^2 d + d \log d, 0, 0, 0, 0)$.

4.3.2 Else do the following.

4.3.2.1 For $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(\ell^2 d^2 \log^2 d \,\mathrm{mpe}(q-1)^2 + \ell d^3 \,\mathrm{mpe}(q-1)^2 \log q, 0, 0, 0, 0)$.

4.3.2.1.1 If $h < H_i$, then set $h' := h$; otherwise, set $h' := \mathcal{H}_{i_t}$.
QC: $(\log d + \log\log q, 0, 0, 0, 0)$.

4.3.2.1.2 For $k = 0, 1, \ldots, h' - 1$, set

$$w_k := \begin{cases} \mathrm{proc}_{i_t,k+1} - \mathrm{proc}_{i_t,k+2}, & \text{if } h = H_i, \text{ or } h < H_i \text{ and } k < h' - 1, \\ \mathrm{proc}_{i_t,h}, & \text{if } h < H_i \text{ and } k = h' - 1. \end{cases}$$

QC: $(\ell \,\mathrm{mpe}(q-1) \log q, 0, 0, 0, 0)$.

4.3.2.1.3 For $k = 0, 1, \ldots, h' - 1$, find $\bar{n}_{i_t,k}$, the unique $n \in \mathcal{N}$ such that $i_t \in S_{n,\mathrm{trans}}$ and $\mathrm{ht}_n = k$.
QC: $(\ell d^2 \log^2 d \operatorname{mpe}(q-1)^2 + \ell \operatorname{mpe}(q-1) \log\log q, 0, 0, 0, 0)$.

4.3.2.1.4 Set $\mathfrak{D} := \{(\bar{n}_{i_t,k}, w_k) : k = 0, 1, \ldots, h' - 1\}$, sorted lexicographically.
QC: $(\ell \operatorname{mpe}(q-1)(\log \ell + \log \operatorname{mpe}(q-1)) \log q, 0, 0, 0, 0)$.

4.3.2.1.5 Check whether $\mathfrak{D} = \mathfrak{D}_n$ some (unique) $n \in \mathcal{N}$, and if so, store this information (the truth value and n).
QC: $(\ell d^2 \operatorname{mpe}(q-1)^2 \log q, 0, 0, 0, 0)$.

4.3.2.1.6 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

4.3.2.2.6.1 If $h < H_i$, then merge the sorted lists $\{i_t\}$ and $S_{n,\mathrm{trans}}$. Otherwise, merge the sorted lists $\{i_t\}$ and $S_{n,\mathrm{per}}$.
QC: $(d \log d, 0, 0, 0, 0)$.

4.3.2.1.7 Else do the following.

(1) Set $n' := \max \mathcal{N} + 1$, $\mathfrak{D}_{n'} := \mathfrak{D}$, $\mathrm{ht}_{n'} := h'$, and add n' to $\mathcal{N}$ as a new element.
QC: $(d \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

(2) Initialize $S_{n',\mathrm{trans}} := \emptyset$ and $S_{n',\mathrm{per}} := \emptyset$.
QC: $(\log d + \log\log q, 0, 0, 0, 0)$

(3) If $h < H_i$, then add i_t to $S_{n',\mathrm{trans}}$ as a new element. Otherwise, add i_t to $S_{n',\mathrm{per}}$ as a new element.
QC: $(\log d + \log\log q, 0, 0, 0, 0)$.

5 Compute $\mathfrak{H} := \max\{H_i : (i, \ell) \in \bar{\mathcal{L}}\}$.
QC: $(d(\log d + \log\log q), 0, 0, 0, 0)$.

6 For $n \in \mathcal{N}$, do the following.
QC: $(d^3 \log d \operatorname{mpe}(q-1) + d^2 \operatorname{mpe}(q-1) \log\log q, 0, 0, 0, 0)$.

6.1 Set $S_n := (\mathrm{ht}_n, S_{n,\mathrm{trans}}, S_{n,\mathrm{per}})$.
QC: $(d \log d + \log\log q, 0, 0, 0, 0)$.

7 Output $\mathfrak{R} := ((\mathfrak{D}_n, S_n))_{n \in \mathcal{N}}$ and halt.
QC: $(d^4 \operatorname{mpe}(q-1)^2 \log q, 0, 0, 0, 0)$.

Next, we give pseudocode for the algorithm from Proposition 5.3.2.3, which computes the cycle type $\mathrm{CT}(A_{|\mathrm{per}(A)})$ for a given affine map $A : x \mapsto ax + b$ of $\mathbb{Z}/m\mathbb{Z}$.

1 Factor m, compute $\mathrm{ord}_{p^{\nu_p(m)}}(a)$ for all primes $p \mid m$ with $p \nmid a$, factor $a - 1$, and if $\nu_2(m) \geq 2$, compute $\varepsilon \in \{0, 1\}$ and $e \in \{0, 1, \ldots, 2^{\nu_2(m)} - 2\}$ such that $a \equiv (-1)^\varepsilon 5^e \pmod{2^{\nu_2(m)}}$.
QC: $(\log m, 0, 1, 1, 0)$.

2 For each prime $p \mid m$ such that $p \nmid a$, do the following.
QC: $(\log^{2+o(1)} m, 0, \log m, 0, 0)$.

2.1 Compute $A_{(p)} := A \bmod p^{\nu_p(m)}$.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

2.2 Using [15, Tables 3 and 4], compute

$$\mathrm{CT}(A_{(p)}) = x_{\bar{l}_{p,1}}^{e_{p,1}} x_{\bar{l}_{p,2}}^{e_{p,2}} \cdots x_{\bar{l}_{p,K_p}}^{e_{p,K_p}} \quad (\text{all } e_{p,j} > 0)$$

with all cycle lengths $\bar{l}_{p,j}$ fully factored. This involves factoring $\mathrm{ord}_{p^{\nu_p(m)}}(a)$, a single power computation, and $O(\nu_p(m))$ instances of simpler arithmetic.
QC: $(\log^{2+o(1)} p^{\nu_p(m)} + \nu_p(m) \log^{1+o(1)} p^{\nu_p(m)}, 0, 1, 0, 0)$.

3 For each $\vec{j} = (j_p)_{p|m, p\nmid a} \in \prod_{p|m,p\nmid a} \{1, 2, \ldots, K_p\}$, do the following.
QC: $(\tau(m) \log^{2+o(1)} m, 0, 0, 0, 0)$.

3.1 Compute the Wei–Xu product of variable powers

$$\mathrm{WX}(\vec{j}) := \divideontimes_{p|m,p\nmid a} x_{\bar{l}_{p,j_p}}^{e_{p,j_p}} = x_{\mathrm{lcm}(\bar{l}_{p,j_p}:p|m,p\nmid a)}^{\prod_{p|m,p\nmid a}(e_{p,j_p}\bar{l}_{p,j_p})/\mathrm{lcm}(\bar{l}_{p,j_p}:p|m,p\nmid a)}.$$

QC: $(\log^{2+o(1)} m, 0, 0, 0, 0)$.

4 Compute and output

$$\mathrm{CT}(A) = \divideontimes_{p|m,p\nmid a} \mathrm{CT}(A_{(p)}) = \prod_{\vec{j}} \mathrm{WX}(\vec{j}),$$

then halt.
QC: $(\tau(m)^2 \log m, 0, 0, 0, 0)$.

The following is pseudocode for the algorithm from Lemma 5.3.2.7, serving to compute $\mathrm{minperl}(\vec{x})$ for given $\vec{x} \in \{0, 1, \ldots, N-1\}^n$, where each $n \in \{0, 1, \ldots, N-1\}$ is given with bit length l_{bit}.

1 Compute the binary representation of n
QC: $(n, 0, 0, 0, 0)$.

2 Factor $n = p_1^{v_1} \cdots p_K^{v_K}$ deterministically.
QC: $(n^{1/5+o(1)}, 0, 0, 0, 0)$.

3 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(n \log n \log\log n (\log n + l_{\mathrm{bit}}))$.

3.1 Using binary search, find $v'_j = \nu_{p_j}(\mathrm{minperl}(\vec{x}))$ as the smallest

$$v \in \{0, 1, \ldots, v_j\}$$

such that $p_j^v \prod_{k\neq j} p_k^{v_k}$ is a period length of $\vec{x}$.
QC: $(n \log\log n (\log n + l_{\mathrm{bit}}))$.

4 Compute and output

$$\operatorname{minperl}(\vec{x}) = \prod_{j=1}^{K} p_j^{v'_j},$$

then halt.
QC: $(\log^{3+o(1)} n, 0, 0, 0, 0)$.

Next, we give pseudocode for Theorem 5.3.2.10, which is concerned with computing not only a tree register, but also an associated tree necklace list for a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$ that is of special type I or II. Because the procedures for the two cases are analogous, we just give one algorithm that deals with both simultaneously.

1 Check whether f is of special type I and store this information. In the process, also compute and store $\bar{f}$ and the affine maps A_i for later use.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 If f is *not* of special type I, then check whether f is of special type II and store this information.
QC: $(d \log^2 d, 0, 0, 0, 0)$ because $\bar{f}$ has already been computed.

3 If f is neither of special type I nor II, then output "fail" and halt.
QC: $(1, 0, 0, 0, 0)$.

4 If f is of special type I, then do the following.

4.1 Compute a type-I tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d)$.
QC: $(d^3 \log^2 d + d \log^{1+o(1)} q, 0, 0, 0, 0)$ because $\bar{f}$ and the A_i have already been computed.

5 Else do the following.

5.1 Compute a type-II tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with

$$N \in O(d^2 \operatorname{mpe}(q-1)),$$

where $S_n = (\mathrm{ht}_n, S_{n,\mathrm{trans}}, S_{n,\mathrm{per}})$.
QC: $(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^5 \operatorname{mpe}(q-1)^3 \log q + d^5 \log^2 d \operatorname{mpe}(q-1)^3, 0, 0, 0, 0)$ because $\bar{f}$ and the A_i have already been computed.

6 Compute a CRL-list $\bar{\mathfrak{L}}$ of $\bar{f}$ and, in the process, store the cycles of $\bar{f}$.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

7 For each $(i, \ell) \in \bar{\mathfrak{L}}$, with associated $\bar{f}$-cycle $(i_0, i_1, \ldots, i_{\ell-1})$, do the following.
QC: $(d\tau(q-1) \log^{2+o(1)} q + d^2 \log^{1+o(1)} q + d\tau(q-1)^2 \log q + Nd^2 \log d + d^3 \log d\, l_{\mathrm{bit}}, 0, d \log q, d, 0)$.

7.1 If $i = d$, then do the following.

7.1.1 If f is of special type I, then do the following.

7.1.1.1 Set $\mathfrak{n}$ to be the unique $n \in \{0, 1, \ldots, N\}$ such that $d \in S_n$.
QC: $(d \log d, 0, 0, 0, 0)$.

7.1.2 Else do the following.

7.1.2.1 Set $\mathfrak{n} := 0$.
QC: $(1, 0, 0, 0, 0)$.

7.1.3 Set $\mathfrak{N}_d := \{([\mathfrak{n}], 1, 1)\}$, $\vec{\mathfrak{n}}'_d := [\mathfrak{n}]$ and $\mathfrak{N}'_d := \{(1, 1)\}$, then skip to the next pair (i, ℓ).
QC: $(\log d, 0, 0, 0, 0)$.

7.2 Else do the following.

7.2.1 Compute $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

7.2.2 Compute $\mathrm{CT}((\mathcal{A}_i)_{|\mathrm{per}(\mathcal{A}_i)}) = x_1^{e_{i,1}} x_2^{e_{i,2}} \cdots x_s^{e_{i,s}}$.
QC: $(\tau(q-1) \log^{2+o(1)} q + \tau(q-1)^2 \log q, 0, \log q, 1, 0)$.

7.2.3 If f is of special type I, then do the following.

7.2.3.1 For each $n = 0, 1, \ldots, N$, do the following.
QC: $(d^2 \log d, 0, 0, 0, 0)$.

7.2.3.1.1 For each $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(|S_n| d \log d, 0, 0, 0, 0)$.

7.2.3.1.1.1 If $i_t \in S_n$, then set $\mathfrak{n}_{i_t} := n$.
QC: $(|S_n| \log d, 0, 0, 0, 0)$.

7.2.4 Else do the following.

7.2.4.1 For each $n = 0, 1, \ldots, N$, do the following.
QC: $(d^3 \log d \ \mathrm{mpe}(q-1) + d\, l_{\mathrm{bit}}, 0, 0, 0, 0)$.

7.2.4.1.1 For each $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(\max\{1, |S_{n,\mathrm{per}}|\} d \log d + |S_{n,\mathrm{per}} \cap \{i_0, \ldots, i_{\ell-1}\}| l_{\mathrm{bit}}, 0, 0, 0, 0)$.

7.2.4.1.1.1 If $i_t \in S_{n,\mathrm{per}}$, then set $\mathfrak{n}_{i_t} := n$.
QC: $(\max\{1, |S_{n,\mathrm{per}}|\} \log d + l_{\mathrm{bit}}, 0, 0, 0, 0)$.

7.2.5 Set $\vec{\mathfrak{n}}_i := (\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\ell-1}})$.
QC: $(d\, l_{\mathrm{bit}}, 0, 0, 0, 0)$.

7.2.6 Overwrite $\vec{\mathfrak{n}}_i$ with the lexicographically smallest sequence in the same cyclic equivalence class.
QC: $(d^2 \log d\, l_{\mathrm{bit}}, 0, 0, 0, 0)$.

7.2.7 Compute $\mathrm{minperl}(\vec{\mathfrak{n}}_i)$.
QC: $(d \log d \log\log d (\log d + l_{\mathrm{bit}}), 0, 0, 0, 0)$.

7.2.8 Set

- $\vec{\mathfrak{n}}'_i := [\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\text{minperl}(\vec{\mathfrak{n}}_i)-1}}]$,
- $\mathfrak{N}'_i := \{(\ell \cdot l', e_{i,l'}) : l' \in \{1, 2, \ldots, s\}, e_{i,l'} > 0\}$, and
- $\mathfrak{N}_i := \{\vec{\mathfrak{n}}'_i\} \times \mathfrak{N}'_i$.

QC: $(d l_{\text{bit}} + \tau(q-1)\log^{1+o(1)} q, 0, 0, 0, 0)$.

8 Set $\mathfrak{M} := \mathfrak{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

9 Set $\vec{\mathfrak{n}}' := \{(\vec{\mathfrak{n}}'_i, i, \ell) : (i, \ell) \in \overline{\mathcal{L}}\}$, and sort it lexicographically.
QC: $(d^2 \log d l_{\text{bit}}, 0, 0, 0, 0)$.

10 For each $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$, do the following.
QC: $(d^3\tau(q-1)^2 \log q, 0, 0, 0, 0)$.

10.1 Check whether $\vec{\mathfrak{n}}'_i \in \mathfrak{M}$, and if so, skip to the next triple $(\vec{\mathfrak{n}}'_i, i, \ell)$.
QC: $(d^2 l_{\text{bit}}, 0, 0, 0, 0)$.

10.2 Add $\vec{\mathfrak{n}}'_i$ to $\mathfrak{M}$ as a new element.
QC: $(d l_{\text{bit}}, 0, 0, 0, 0)$.

10.3 Set $\mathfrak{V} := \mathfrak{N}'_i$.
QC: $(\tau(q-1)\log q, 0, 0, 0, 0)$.

10.4 For each $(\vec{\mathfrak{n}}'_j, j, \ell') \in \vec{\mathfrak{n}}'$ that comes after $(\vec{\mathfrak{n}}'_i, i, \ell)$, do the following.
QC: $(d^2 l_{\text{bit}} + d^2\tau(q-1)^2 \log q, 0, 0, 0, 0)$.

10.4.1 Check whether $\vec{\mathfrak{n}}'_i = \vec{\mathfrak{n}}'_j$, and if not, skip to the next triple $(\vec{\mathfrak{n}}'_j, j, \ell')$.
QC: $(d l_{\text{bit}}, 0, 0, 0, 0)$.

10.4.2 For each $(l, k) \in \mathfrak{N}'_j$, do the following.
QC: $(d\tau(q-1)^2 \log q, 0, 0, 0, 0)$.

10.4.2.1 Check whether l occurs as the first entry of some pair $(l, k') \in \mathfrak{V}$, and store this information.
QC: $(d\tau(q-1)\log q, 0, 0, 0, 0)$.

10.4.2.2 If $(l, k') \in \mathfrak{V}$ for some k', then do the following.

10.4.2.2.1 Replace the unique element of $\mathfrak{V}$ of the form (l, k') by $(l, k + k')$.
QC: $(\log q, 0, 0, 0, 0)$.

10.4.2.3 Else do the following.

10.4.2.3.1 Add (l, k) to $\mathfrak{V}$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

10.5 For each $(l, k) \in \mathfrak{V}$, do the following.
QC: $(d\tau(q-1)(d l_{\text{bit}} + \log q), 0, 0, 0, 0)$.

10.5.1 Add $([\vec{\mathfrak{n}}'_i], l, k)$ to $\mathfrak{N}$ as a new element.
QC: $(d l_{\text{bit}} + \log q, 0, 0, 0, 0)$.

11 Output $\mathfrak{R}$ and $\mathfrak{N}$, and halt.
QC: $(N^2 \log q + d\tau(q-1)(d\, l_{\text{bit}} + \log q), 0, 0, 0, 0)$.

The following is pseudocode for the algorithm from Lemma 5.3.2.12. For given recursive tree description lists $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$ and $\vec{\mathfrak{D}}' = (\mathfrak{D}'_n)_{n=0,1,\ldots,N'}$ satisfying the assumptions of Lemma 5.3.2.12, this algorithm computes a synchronization $(\vec{\mathfrak{D}}^+, \mathrm{i})$ of $\vec{\mathfrak{D}}$ and $\vec{\mathfrak{D}}'$.

1 Set $\mathfrak{R} := \{0, 1, \ldots, N\}$, and for $k \in \mathfrak{R}$, set $\mathfrak{D}_k^+ := \mathfrak{D}_k$. Moreover, let i be the empty function $\emptyset$.
QC: $(N^2(l_{\text{bit}} + m), 0, 0, 0, 0)$.

2 For each $n = 0, 1, \ldots, N'$, do the following.
QC: $((NN' \min\{N, N'\} + (N')^2 l'_{\text{bit}})(l_{\text{bit}} + l'_{\text{bit}} + m), 0, 0, 0, 0)$.

2.1 If $n = 0$, then do the following.

2.1.1 Set $\mathrm{i}(0) := 0$.
QC: $(1, 0, 0, 0, 0)$.

2.2 Else do the following.

2.2.1 Let $\mathfrak{D}$ be the set of pairs obtained from $\mathfrak{D}'_n$ through replacing each first entry k of each pair in $\mathfrak{D}'_n$ by $\mathrm{i}(k)$ (one may simply overwrite the corresponding entries of $\mathfrak{D}'_n$, so one does not need to handle the second entries of bit length in $O(m)$).
QC: $(N'(l_{\text{bit}} + l'_{\text{bit}}), 0, 0, 0, 0)$.

2.2.2 Sort $\mathfrak{D}$.
QC: $(N' l'_{\text{bit}}(l_{\text{bit}} + l'_{\text{bit}} + m))$.

2.2.3 Check whether $\mathfrak{D} = \mathfrak{D}_k^+$ for some (unique) $k \in \{0, 1, \ldots, N\}$, and store this information (the truth value and k).
QC: $(N \min\{N, N'\}(l_{\text{bit}} + l'_{\text{bit}} + m), 0, 0, 0, 0)$.

2.2.4 If $\mathfrak{D} = \mathfrak{D}_k^+$ for some $k \in \{0, 1, \ldots, N\}$, then do the following.

2.2.4.1 Set $\mathrm{i}(n) := k$.
QC: $(l_{\text{bit}} + l'_{\text{bit}}, 0, 0, 0, 0)$.

2.2.5 Else do the following.

2.2.5.1 Set $n' := \max \mathcal{N} + 1$, add n' to $\mathcal{N}$ as a new element, set $\mathfrak{D}_{n'}^+ := \mathfrak{D}$ and $\mathrm{i}(n) := n'$.
QC: $(\min\{N, N'\}(l_{\text{bit}} + l'_{\text{bit}} + m), 0, 0, 0, 0)$.

3 Set $\vec{\mathfrak{D}}^+ := (\mathfrak{D}_n^+)_{n \in \mathcal{N}}$, output $(\vec{\mathfrak{D}}^+, \mathrm{i})$ and halt.
QC: $((N + N') \max\{N, N'\}(l_{\text{bit}} + l'_{\text{bit}} + m), 0, 0, 0, 0)$.

Finally, we provide pseudocode for the algorithm from Corollary 5.3.2.13. For given generalized cyclotomic mappings f_1 and f_2 of $\mathbb{F}_q$, of index d_1 and d_2, respectively, such that each f_j is of special type I or II (not necessarily both of the same

special type), this algorithm decides whether $\Gamma_{f_1} \cong \Gamma_{f_2}$. Throughout this discussion, we have $d := \max\{d_1, d_2\}$.

1 For $j = 1, 2$, do the following.
QC:
- $(d \log^{1+o(1)} q, d, 0, 0, 0)$ if f_1 and f_2 are both of special type I;
- $(d \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0)$ otherwise.

1.1 Check whether f_j is of special type I, and store this information as well as the induced function $\overline{f_j}$ and the affine maps on $\mathbb{Z}/((q-1)/d_j)\mathbb{Z}$ associated with f_j.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

1.2 If f_j is *not* of special type I, then check whether f_j is of special type II, and store this information.
QC: $(d \log^2 d, 0, 0, 0, 0)$, because $\overline{f_j}$ and the affine maps have already been computed.

1.3 If f_j is neither of special type I nor II, then output "fail" and halt.
QC: $(1, 0, 0, 0, 0)$.

2 For $j = 1, 2$, do the following.
QC:
- $(d^3 \log^2 d + d^3 \log d\,\tau(q-1) + d^3\tau(q-1)^2 \log q + d^2 \log^{1+o(1)} q + d\,\tau(q-1) \log^{2+o(1)} q, d, d \log q, d, 0)$ if f_1 and f_2 are both of special type I;
- $(d^5 \log^2 d\,\mathrm{mpe}(q-1) + (d^5\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2) \log q + d^2\,\mathrm{mpe}(q-1) \log^{1+o(1)} q + d\,\tau(q-1) \log^{2+o(1)} q, d, d \log q, d, 0)$ otherwise.

2.1 If f_j is of special type I, then do the following.

2.1.1 Compute a type-I tree register $\Re_j$ of f_j, and the tree necklace list $\Re_j$ for f_j relative to $\Re_j$.
QC: $(d^3 \log^2 d + d^3\tau(q-1)^2 \log q + d^2 \log^{1+o(1)} q + d\,\tau(q-1) \log^{2+o(1)} q, d, d \log q, d, 0)$.

2.2 Else do the following.

2.2.1 Compute a type-II tree register $\Re_j$ of f_j, and the tree necklace list $\Re_j$ for f_j relative to $\Re_j$.
QC: $(d^5 \log^2 d\,\mathrm{mpe}(q-1)^3 + (d^5\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2) \log q + d^2\,\mathrm{mpe}(q-1) \log^{1+o(1)} q + d\,\tau(q-1) \log^{2+o(1)} q, d, d \log q, d, 0)$.

3 For $j = 1, 2$, let $\vec{\mathfrak{D}}^{(j)} = (\mathfrak{D}_n^{(j)})_{n=0,1,\dots,N_j}$ be the underlying recursive tree description list of $\mathfrak{R}_j$. Compute a synchronization $(\vec{\mathfrak{D}}^+, \mathfrak{i})$ of $\vec{\mathfrak{D}}^{(1)}$ and $\vec{\mathfrak{D}}^{(2)}$.
QC:
- $(d^3 \log q, 0, 0, 0, 0)$ if f_1 and f_2 are both of special type I;
- $(d^6 \operatorname{mpe}(q-1)^3 \log q + d^4 \operatorname{mpe}(q-1)^2 \log^{1+o(1)} q, 0, 0, 0, 0)$ otherwise.

4 Create a modified version $\mathfrak{R}_2'$ of $\mathfrak{R}_2$ by replacing each entry $\mathfrak{n}$ of each first entry $[\vec{\mathfrak{n}}]$ of an element $([\vec{\mathfrak{n}}], l, \mathfrak{m}) \in \mathfrak{R}_2$ by $\mathfrak{i}(\mathfrak{n})$, then overwriting each of the resulting $O(d)$ distinct first entries of triples in the list with the lexicographically minimal number sequence in the same cyclic equivalence class, and finally sorting $\mathfrak{R}_2'$ lexicographically by merging the $O(d)$ distinct segments corresponding to the same first entry of triples in $\mathfrak{R}_2'$.
QC: $(d^3 \tau(q-1) \log q, 0, 0, 0, 0)$.

5 Check whether $\mathfrak{R}_2' = \mathfrak{R}_1$, output the corresponding truth value, and halt.
QC: $(d^2 \tau(q-1) \log q, 0, 0, 0, 0)$.

5.3.3 Short-term block behavior and the special case where all cycles are short

Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$. For an f-periodic $x \in \mathbb{F}_q$ and $t \in \mathbb{Z}$, we set $x^{(t)} := (f_{|\mathrm{per}(f)})^t(x)$, and we recall the notation $i_t := (\bar{f}_{|\mathrm{per}(\bar{f})})^t(i)$ for $\bar{f}$-periodic $i \in \{0, 1, \dots, d-1\}$ and $t \in \mathbb{Z}$, as well as $i' := i_{-1}$.

In Section 3.3, for each $i \in \{0, 1, \dots, d-1\}$, we constructed an arithmetic partition $\mathcal{P}_i$ of C_i such that for $x \in C_i$, the isomorphism type of $\mathrm{Tree}_{\Gamma_f}(x)$ only depends on the $\mathcal{P}_i$-block in which x is contained. Here, we refine this construction. We recall that $\mathcal{P}_i = \mathcal{Q}_{i,H_i}$, where H_i is the maximum tree height in Γ_{per} (the induced subgraph of Γ_f on $\bigcup_{j \in \mathrm{per}(\bar{f})} C_j$) above an f-periodic point in $\bigcup_{t \in \mathbb{Z}} C_{i_t}$. The arithmetic partition $\mathcal{Q}_{i,h}$ of C_i is defined for all $h \in \mathbb{N}_0$ (even though we only considered it for $h \in \{0, 1, \dots, H_i\}$ in Section 3.3) and satisfies

$$\mathcal{Q}_{i,h+1} = \mathcal{R}_i \wedge \mathfrak{P}'(\mathcal{Q}_{i',h}, A_{i'}). \tag{5.7}$$

This formula is key to our construction. Indeed, the refined arithmetic partition of C_i which we consider here is simply $\mathcal{Q}_{i,H_i+L-1}$ for some $L \in \mathbb{N}^+$, as opposed to $\mathcal{P}_i = \mathcal{Q}_{i,H_i}$. While the blocks of $\mathcal{P}_i$ control the isomorphism types of rooted trees in Γ_f above (periodic) vertices in C_i, the following more general statement holds for $\mathcal{Q}_{i,H_i+L-1}$.

Lemma 5.3.3.1. *Let $x \in C_i$ be f-periodic, and let $L \in \mathbb{N}_0$. The $\mathcal{Q}_{i,H_i+L-1}$-block in which x is contained uniquely determines the (length L) sequence*

$$(\mathrm{Tree}_{\Gamma_f}(x^{(t)}))_{t=0,-1,\dots,-L+1}$$

of rooted tree isomorphism types.

Proof. We can write the $\mathcal{Q}_{i,H_i+L-1}$-block of x as $\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i})$, where $\vec{o'_t} \in \{\emptyset, \neg\}^{n_{i-t}}$. Noting that

$$\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t} \diamond \vec{\xi}_{i,H_i}) \subseteq \mathcal{B}(\mathcal{P}_i, \diamond_{t=0}^{H_i}\vec{o'_t} \diamond \vec{\xi}_{i,H_i}),$$

we see that $\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_i(\mathcal{P}_i, \diamond_{t=0}^{H_i}\vec{o'_t} \diamond \vec{\xi}_{i,H_i})$ is uniquely determined.

By formula (5.7), we know that for each block $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \diamond_{t=0}^{H_i+L-2}\vec{o_t} \diamond \vec{\xi})$ of $\mathcal{Q}_{i',H_i+L-2}$, the number of f-pre-images of x in that block is the constant

$$\sigma_{\mathcal{Q}_{i',H_i+L-2},A_{i'}}(\diamond_{t=0}^{H_i+L-2}\vec{o_t} \diamond \vec{\xi}, \diamond_{t=1}^{H_i+L-1}\vec{o'_t} \diamond \vec{\xi}_{i,H_i}). \tag{5.8}$$

Now we assume that $\vec{\xi} = \vec{\xi}_{i',H_i}$ (which is actually the same as $\vec{\xi}_{i,H_i}$). The union of all blocks of $\mathcal{Q}_{i',H_i+L-2}$ of the form $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \diamond_{t=0}^{H_i+L-2}\vec{o_t} \diamond \vec{\xi}_{i',H_i})$ (where $\vec{o_t}$ ranges over $\{\emptyset, \neg\}^{n_{i-t-1}}$ for each $t \in \{0, 1, \ldots, H_i + L - 2\}$) is just the subset of $C_{i'}$ consisting of all f-periodic points in it. Since x has precisely one f-periodic pre-image (which lies in $C_{i'}$), it follows that the value of (5.8) for $\vec{\xi} = \vec{\xi}_{i',H_i}$ is 0 for all $\diamond_{t=0}^{H_i+L-2}\vec{o_t} \in \{\emptyset, \neg\}^{n_{i-1}+n_{i-2}+\cdots+n_{i-H_i-L+1}}$ except for one, for which the constant (5.8) has value 1. If $\diamond_{t=0}^{H_i+L-2}\vec{o_t}$ is that unique logical sign tuple, then the unique f-periodic pre-image $x^{(-1)}$ of $x \in \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t} \diamond \vec{\xi}_{i,H_i})$ always lies in $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \diamond_{t=0}^{H_i+L-2}\vec{o_t} \diamond \vec{\xi}_{i',H_i})$, and so

$$\mathrm{Tree}_{\Gamma_f}(x^{(-1)}) \cong \mathrm{Tree}_{i'}(\mathcal{P}_{i'}, \diamond_{t=0}^{H_i}\vec{o_t} \diamond \vec{\xi}_{i',H_i})$$

is also uniquely determined. Continuing this process inductively, we get the statement of the lemma. ∎

For the purposes of our later complexity analysis, we need a more explicit version of Lemma 5.3.3.1. To each $\bar{f}$-periodic $i \in \{0, 1, \ldots, d-1\}$ and each $L \in \mathbb{N}^+$, we associate the set

$$\mathcal{O}_{i,L} := \{\diamond_{t=0}^{H_i+L-1}\vec{o'_t} \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-H_i-L+1}}} : \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t} \diamond \vec{\xi}_{i,H_i}) \neq \emptyset\}$$

of logical sign tuples that correspond to a non-empty block of $\mathcal{Q}_{i,H_i+L-1}$ consisting of f-periodic points. The proof of Lemma 5.3.3.1 shows that as long as $L \geq 2$, we may implicitly define a (surjective) function $\mathfrak{u}_{i,L} : \mathcal{O}_{i,L} \to \mathcal{O}_{i',L-1}$ via

$$\sigma_{\mathcal{Q}_{i',H_i+L-2},A_{i'}}(\mathfrak{u}_{i,L}(\diamond_{t=0}^{H_i+L-1}\vec{o'_t}) \diamond \vec{\xi}_{i',H_i}, \diamond_{t=1}^{H_i+L-1}\vec{o'_t} \diamond \vec{\xi}_{i,H_i}) = 1.$$

Then for each (f-periodic) $x \in \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t} \diamond \vec{\xi}_{i,H_i})$, the unique f-periodic pre-image $x^{(-1)}$ of x, which lies in $C_{i'}$, is contained in

$$\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \mathfrak{u}_{i,L}(\diamond_{t=0}^{H_i+L-1}\vec{o'_t}) \diamond \vec{\xi}_{i',H_i}).$$

Denoting by $\mathrm{proj}_{i,L}$ the projection

$$\mathcal{O}_{i,L} \to \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-H_i}}}, \quad \diamond_{t=0}^{H_i+L-1}\vec{o}'_t \mapsto \diamond_{t=0}^{H_i}\vec{o}'_t,$$

we therefore have the following more explicit version of Lemma 5.3.3.1.

Lemma 5.3.3.2. *Let $L \in \mathbb{N}^+$, and let $x \in C_i$ be f-periodic, say contained in*

$$\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o}'_t).$$

Then for each $k = 0, -1, \ldots, -L+1$, we have

$$\mathrm{Tree}_{\Gamma_f}(x^{(k)}) \cong$$
$$\mathrm{Tree}_{i_k}(\mathcal{P}_{i_k}, (\mathrm{proj}_{i_k,L+k} \circ \mathfrak{u}_{i_{k+1},L+k+1} \circ \mathfrak{u}_{i_{k+2},L+k+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{t=0}^{H_i+L-1}\vec{o}'_t)).$$

We note that the composition of functions of the form $\mathfrak{u}_{i_t,L+t}$ in the formula in Lemma 5.3.3.2 is empty if $k = 0$ (index-wise, it is supposed to ascend from 1 to 0, which is nonsensical). Specifically, Lemma 5.3.3.2 for $k = 0$ states that

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}(x^{(0)}) = \mathrm{Tree}_{i_0}(\mathcal{P}_{i_0}, \mathrm{proj}_{i_0,L}(\diamond_{t=0}^{H_i+L-1}\vec{o}'_t)),$$

for $k = -1$, it states that

$$\mathrm{Tree}_{\Gamma_f}(x^{(-1)}) = \mathrm{Tree}_{i_{-1}}(\mathcal{P}_{i_{-1}}, \mathrm{proj}_{i_{-1},L-1} \circ \mathfrak{u}_{i_0,L}(\diamond_{t=0}^{H_i+L-1}\vec{o}'_t)),$$

and so on.

In what follows, let us assume that all cycle lengths of f are at most L. We consider an $\bar{f}$-periodic index i of cycle length ℓ. By Lemma 5.3.3.2, for each $x \in C_i$, the block of $\mathcal{Q}_{i,H_i+L-1}$ in which x is contained together with the precise f-cycle length of x completely determines the digraph isomorphism type of the connected component of Γ_f containing x. By adding suitable s-congruences to the spanning congruences of $\mathcal{Q}_{i,H_i+L-1}$, we can construct a finer arithmetic partition, denoted by $\mathcal{W}_{i,L}$ below, each block of which consists of points of a common f-cycle length. Hence, for each f-periodic point $x \in C_i$, the digraph isomorphism type of the connected component of Γ_f containing x is completely determined by the $\mathcal{W}_{i,L}$-block containing x.

Let us discuss the details of how to construct $\mathcal{W}_{i,L}$. We recall from Section 3.3 that for each $l \in \mathbb{N}^+$, the restriction of f^l to $C_{i_{-l}}$, which maps to C_i, is represented by the affine map $\mathcal{A}_{i,l} : x \mapsto \bar{\alpha}_{i,l}x + \bar{\beta}_{i,l}$ (formulas for $\bar{\alpha}_{i,l}$ and $\bar{\beta}_{i,l}$ are given in the first bullet point after Proposition 3.3.3). Therefore, a point $x \in C_i$, viewed as an element of $\mathbb{Z}/s\mathbb{Z}$, is a fixed point of f^l if and only if ℓ divides l (so that $i_{-l} = i$) and $\bar{\alpha}_{i,l}x + \bar{\beta}_{i,l} \equiv x \pmod{s}$. This congruence is solvable if and only if $\gcd(s, \bar{\alpha}_{i,l} - 1) \mid \bar{\beta}_{i,l}$, in which case it is equivalent to the s-congruence

$$x \equiv -\frac{\bar{\beta}_{i,l}}{\gcd(s, \bar{\alpha}_{i,l}-1)} \cdot \mathrm{inv}_{\frac{s}{\gcd(s,\bar{\alpha}_{i,l}-1)}}\left(\frac{\bar{\alpha}_{i,l}-1}{\gcd(s, \bar{\alpha}_{i,l}-1)}\right) \left(\mathrm{mod}\ \frac{s}{\gcd(s, \bar{\alpha}_{i,l}-1)}\right),$$

which we henceforth denote by $\eta_{i,l}(x)$. We observe that $\eta_{i,l}(x)$ is only well defined when $\gcd(s, \overline{\alpha}_{i,l} - 1) \mid \overline{\beta}_{i,l}$. Let us set

$$\mathfrak{C}_{i,L} := \{l \in \{1, 2, \ldots, L\} : \ell \mid l \text{ and } \gcd(s, \overline{\alpha}_{i,l} - 1) \mid \overline{\beta}_{i,l}\}$$

and define

$$\mathcal{V}_{i,L} := \mathfrak{P}(\eta_{i,l}(x) : l \in \mathfrak{C}_{i,L}) \text{ and } \mathcal{W}_{i,L} := \mathcal{Q}_{i,H_i+L-1} \wedge \mathcal{V}_{i,L}.$$

Viewing $\mathcal{V}_{i,L}$ as an arithmetic partition of C_i, we claim that its blocks are just those subsets of C_i that consist of all points of any given f-cycle length. Indeed, let $l \in \{1, 2, \ldots, L\}$. If $l \notin \mathfrak{C}_{i,L}$, then f^l has no fixed points in C_i and, in particular, f has no points of cycle length l in C_i. On the other hand, if $l \in \mathfrak{C}_{i,L}$, then the points $x \in C_i$ of f-cycle length exactly l (if any) are just those that satisfy the congruence $\eta_{i,l'}(x)$ for precisely those $l' \in \mathfrak{C}_{i,L}$ that are multiples of l. In other words, if for $l' \in \mathfrak{C}_{i,L}$ we set

$$\nu_{l,l'} := \begin{cases} \emptyset, & \text{if } l \mid l', \\ \neg, & \text{otherwise,} \end{cases}$$

and set $\vec{\nu}_{i,L,l} := (\nu_{l,l'})_{l' \in \mathfrak{C}_{i,L}}$, then the set $\mathcal{B}(\mathcal{V}_{i,L}, \vec{\nu}_{i,L,l})$ (which may be empty) consists precisely of those $x \in C_i$ that are of f-cycle length l. In summary, we obtain the following result.

Proposition 5.3.3.3. *Let $L \in \mathbb{N}^+$ be such that all cycle lengths of f are at most L, and let $i \in \{0, 1, \ldots, d-1\}$ be $\bar{f}$-periodic. We view $\mathcal{V}_{i,L}$ and $\mathcal{W}_{i,L}$ as arithmetic partitions of C_i. Then the following hold.*

(1) *Each block of $\mathcal{V}_{i,L}$ is of one of the forms $\mathcal{B}(\mathcal{V}_{i,L}, (\neg, \neg, \ldots, \neg))$, respectively, $\mathcal{B}(\mathcal{V}_{i,L}, \vec{\nu}_{i,L,l})$ for some $l \in \mathfrak{C}_{i,L}$, and it consists precisely of the f-transient points in C_i, respectively of those f-periodic points in C_i that have f-cycle length precisely l.*

(2) *Each block of $\mathcal{W}_{i,L}$ consists either entirely of f-periodic or entirely of f-transient points. Moreover, each block of $\mathcal{W}_{i,L}$ whose elements are f-periodic is of the form*

$$\mathcal{B}(\mathcal{W}_{i,L}, \diamond_{t=0}^{H_i+L-1} \vec{o}'_t \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l})$$

for some $\vec{o}'_t \in \{\emptyset, \neg\}^{n_{i-t}}$ and $l \in \mathfrak{C}_{i,L}$, in which case for any given point x in that block, the digraph isomorphism type of the connected component of Γ_f containing x is represented by the cyclic sequence

$$[\mathrm{Tree}_{i_k}(\mathcal{P}_{i_k}, (\mathrm{proj}_{i_k,L+k} \circ \mathfrak{u}_{i_{k+1},L+k+1} \circ \mathfrak{u}_{i_{k+2},L+k+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{t=0}^{H_i+L} \vec{o}'_t))]_{k=-l+1,-l+2,\ldots,0}$$

of rooted tree isomorphism types.

Proposition 5.3.3.3 is the basis for proving the following theorem.

Theorem 5.3.3.4. *Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$. Moreover, let $L \in \mathbb{N}^+$ with $L \leq q-1$ be such that all cycle lengths of f are at most L. Then, within q-bounded query complexity*

$$(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{d \operatorname{mpe}(q-1)} (d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q, d, 0, 0, 0),$$

and thus within q-bounded Las Vegas dual complexity

$$(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{d \operatorname{mpe}(q-1)} (d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q + d \log^{3+o(1)} q, d \log^{3+o(1)} q, d \log q),$$

one can compute

- *a recursive tree description list*

$$\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$$

  with $N \in O(d 2^{d^2 \operatorname{mpe}(q-1)+d})$, whose associated sequence of rooted tree isomorphism types is denoted by $\vec{\mathfrak{T}} = (\mathfrak{T}_n)_{n=0,1,\ldots,N}$, such that indices $n \in \{0, 1, \ldots, N\}$ as well as second entries of elements of a description $\mathfrak{D}_n$ are represented by bit strings of length $l_{\text{bit}} := \lfloor \log_2 q \rfloor + 1$; and
- *the tree necklace list $\mathfrak{N}$ of f relative to $\vec{\mathfrak{T}}$, in the sense of Definition 5.3.2.8, in lexicographically sorted form, which has $O(dL 2^{d^2 \operatorname{mpe}(q-1)+dL})$ distinct elements (triples) and, by convention,*
  - *has the first entries $[\vec{\mathfrak{n}}] = [\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$ of its elements padded analogously to Remark 5.3.2.9 (3), but with $L - l'$ dummy entries -1, so that the bit string representation of $[\vec{\mathfrak{n}}]$ always has the length*

$$L(l_{\text{bit}} + 1) = L(\lfloor \log_2 q \rfloor + 2);$$

  - *uses $\lfloor \log_2 L \rfloor + 1$ bits to represent the second entries of its elements; and*
  - *uses $\lfloor \log_2 q \rfloor + 1$ bits for the third entries of its elements.*

Proof. First, we compute $\bar{f}$, the affine maps A_i and a partition-tree register

$$((\mathcal{Z}_i)_{i=0,1,\ldots,d-1}, ((\mathfrak{D}_n, (S_{n,i})_{i=0,1,\ldots,d}))_{n=0,1,\ldots,N})$$

for f with $N \in O(d 2^{d^2 \operatorname{mpe}(q-1)+d})$; the desired recursive tree description list $\vec{\mathfrak{D}}$ is a part of this. By Proposition 5.1.8 and Theorem 5.1.9 (2), these computations can be carried out within q-bounded query complexity

$$(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d) \operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q, d, 0, 0, 0),$$

which is majorized by the asserted overall q-bounded query complexity for computing $\vec{\mathfrak{D}}$ and $\mathfrak{N}$ (it is this term which necessitates the inclusion of the factor $2^{d\,\mathrm{mpe}(q-1)}$ in the bound on the overall bit operation cost). Moreover, by the proof of Theorem 5.1.9 (2), the following are computed (and may be stored) as part of this:

- the cycles of $\bar{f}$ and a CRL-list $\bar{\mathfrak{L}}$ of $\bar{f}$;
- the parameter H_i for each $\bar{f}$-periodic index $i \in \{0, 1, \ldots, d-1\}$.

Until further notice, we assume that $(i, \ell) \in \bar{\mathfrak{L}}$ with $i < d$ is fixed (the case $i = d$ is easy to deal with separately and will be "tacked on" at the end of this proof). As usual, we let $(i_0, i_1, \ldots, i_{\ell-1})$ with $i = i_0$ be the $\bar{f}$-cycle of i, and set $i_t := i_{t \bmod \ell}$ for arbitrary $t \in \mathbb{Z}$. We analyze the bit operation cost of counting the isomorphism types of connected components of Γ_f that intersect $\bigcup_{t=0}^{d-1} C_{i_t}$ (i.e., that may be represented by a periodic vertex in one of the cosets C_{i_t}). We note that for each $t \in \{0, 1, \ldots, \ell-1\}$, one can directly read off a spanning congruence sequence for $\mathcal{U}_{i_t}$, of length $H_i \leq d\,\mathrm{mpe}(q-1)$, from the partition-tree register computed above. To proceed, we need to determine a spanning congruence sequence of $\mathcal{Q}_{i_{-t}, H_i+L-t-1}$ for $t = 0, 1, \ldots, L-1$. By the definitions of $\mathcal{R}_i$ and $\mathcal{Q}_{i,h}$ from Section 3.3, we have

$$\mathcal{Q}_{i_{-t}, H_i+L-t-1} = \mathcal{R}_{i_{-t}} \wedge \mathfrak{P}'(\mathcal{Q}_{i_{-t-1}, H_i+L-t-2}, A_{i_{-t-1}}).$$

Moreover, we observe that

$$\begin{aligned}
&\mathfrak{P}'(\mathcal{Q}_{i_{-t-1}, H_i+L-t-2}, A_{i_{-t-1}}) \\
&\quad = \mathfrak{P}'\left(\bigwedge_{j=0}^{H_i+L-t-2} \lambda^j_{i_{-t-1-j}}(\mathcal{R}_{i_{-t-1-j}}) \wedge \mathcal{U}_{i_{-t-1}}, A_{i_{-t-1}} \right) \\
&\quad = \bigwedge_{j=0}^{H_i+L-t-2} \lambda^{j+1}_{i_{-t-1-j}}(\mathcal{R}_{i_{-t-1-j}}) \wedge \mathcal{U}_{i_{-t}} \\
&\quad = \bigwedge_{j=1}^{H_i+L-t-1} \lambda^{j}_{i_{-t-j}}(\mathcal{R}_{i_{-t-j}}) \wedge \mathcal{U}_{i_{-t}},
\end{aligned}$$

whence

$$\mathcal{Q}_{i_{-t}, H_i+L-t-1} = \bigwedge_{j=0}^{H_i+L-t-1} \lambda^j_{i_{-t-j}}(\mathcal{R}_{i_{-t-j}}) \wedge \mathcal{U}_{i_{-t}}.$$

Now, from our partition-tree register, we can directly read off a spanning congruence sequence for $\lambda^j_{i_t}(\mathcal{R}_{i_t})$, of length $n_{i_t} \leq d$, for each $t = 0, 1, \ldots, \ell-1$ and each $j = 0, 1, \ldots, H_i$, which altogether takes only $O(d^3\,\mathrm{mpe}(q-1)\log q)$ bit operations for copying. Moreover, for any fixed $t \in \{0, 1, \ldots, \ell-1\}$, we can compute a spanning congruence sequence for

$$\lambda^j_{i_t}(\mathcal{R}_{i_t}) = \lambda(\lambda^{j-1}_{i_t}(\mathcal{R}_{i_t}), A_{i_{t+j-1}})$$

successively for $j = H_i + 1, H_i + 2, \ldots, H_i + L - 1$. For each given t and j, this takes $O(d \log^{1+o(1)} q)$ bit operations, and thus for all t and j together, it takes $O(d^2 L \log^{1+o(1)} q)$ bit operations. Once all of these spanning congruence sequences have been computed, one can paste together such a sequence for a single partition of the form $\mathcal{Q}_{i_{-t}, H_i+L-t-1}$ using $O((H_i + L)d \log q) \subseteq O((d^2 \operatorname{mpe}(q-1) + dL) \log q)$ bit operations. Doing so for all $t = 0, 1, \ldots, L-1$ takes $O((d^2 L \operatorname{mpe}(q-1) + dL^2) \log q)$ bit operations.

Our next goal is to compute the function

$$\mathfrak{u}_{i_{-t}, L-t} : \mathcal{O}_{i_{-t}, L-t} \to \mathcal{O}_{i_{-t-1}, L-t-1}$$

for $t = 0, 1, \ldots, L-2$. To that end, we first compute the set

$$\mathcal{O}_{i_{-t}, L-t} = \{\diamond_{k=0}^{H_i+L-t-1} \vec{o}'_k \in \{\emptyset, \neg\}^{n_{i_{-t}} + n_{i_{-t+1}} + \cdots + n_{i_{-H_i-L+1}}} : \mathcal{B}(\mathcal{Q}_{i_{-t}, H_i+L-t-1}, \diamond_{k=0}^{H_i+L-t-1} \vec{o}'_k \diamond \vec{\xi}_{i_{-t}, H_i}) \neq \emptyset\}$$

for $t = 0, 1, \ldots, L-1$. To do so, we go through the

$$O(2^{d(H_i+L-t)}) \subseteq O(2^{d^2 \operatorname{mpe}(q-1) + dL - dt})$$

tuples $\diamond_{k=0}^{H_i+L-t-1} \vec{o}'_k$, and for each of them, we compute the cardinality of the block

$$\mathcal{B}(\mathcal{Q}_{i_{-t}, H_i+L-t-1}, \diamond_{k=0}^{H_i+L-t-1} \vec{o}'_k \diamond \vec{\xi}_{i_{-t}, H_i}).$$

Following the ideas leading to Proposition 3.3.2, this cardinality is equal to the distribution number

$$\sigma_{\mathcal{Q}_{i_{-t}, H_i+L-t-1}, \mathbf{0}}(\diamond_{k=0}^{H_i+L-t-1} \vec{o}'_k \diamond \vec{\xi}_{i_{-t}, H_i}, (\emptyset, \ldots, \emptyset)) \tag{5.9}$$

of $\mathcal{Q}_{i_{-t}, H_i+L-t-1}$ under the constantly zero affine function $\mathbf{0}$. By the proof of Lemma 5.2.2.1 and the facts that

- the number of spanning congruences of $\mathcal{Q}_{i_{-t}, H_i+L-t-1}$ we are using is at most $d(H_i + L) + H_i \leq d^2 \operatorname{mpe}(q-1) + d \operatorname{mpe}(q-1) + dL \in O(d^2 \operatorname{mpe}(q-1) + dL)$,
- the subsets J we need to loop over never contain any index corresponding to a logical sign for one of the H_i spanning congruences of $\mathcal{U}_{i_{-t}}$, because $\vec{\xi}_{i_{-t}, H_i}$ consists only of positive logical signs, and
- $d(H_i + L) \leq d^2 \operatorname{mpe}(q-1) + dL$,

we conclude that the complexity of computing the distribution number (5.9) is in

$$O(2^{d^2 \operatorname{mpe}(q-1) + dL} (d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q). \tag{5.10}$$

In summary, computing all sets $\mathcal{O}_{i_{-t},L-t}$ for $t = 0, 1, \ldots, L-1$ takes

$$O\left(\sum_{t=0}^{L-1} 4^{d^2\,\mathrm{mpe}(q-1)+dL} 2^{-dt}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)} q\right)$$
$$\subseteq O(4^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)} q)$$

bit operations.

Concerning the computation of the functions $\mathfrak{u}_{i_{-t},L-t}$ themselves, we note that for a given t and argument $\diamond_{k=0}^{H_i+L-t-1}\vec{o}'_k$ of $\mathfrak{u}_{i_{-t},L-t}$, the associated function value $\mathfrak{u}_{i_{-t},L-t}(\diamond_{k=0}^{H_i+L-t-1}\vec{o}'_k)$ is the unique tuple $\diamond_{k=0}^{H_i+L-t-2}\vec{o}_k \in \mathcal{O}_{i_{-t-1},L-t-1}$ such that the distribution number

$$\sigma_{\mathcal{Q}_{i_{-t-1},H_i+L-t-2},A_{i_{-t-1}}}(\diamond_{k=0}^{H_i+L-t-2}\vec{o}_k \diamond \vec{\xi}_{i_{-t-1},H_i}, \diamond_{k=1}^{H_i+L-t-1}\vec{o}'_k \diamond \vec{\xi}_{i_{-t},H_i})$$

is equal to 1 (and, according to the proof of Lemma 5.3.3.1, that distribution number is 0 for all other choices of $\diamond_{k=0}^{H_i+L-t-2}\vec{o}_k$). Therefore, in order to compute each value of every function $\mathfrak{u}_{i_{-t},L-t}$ for $t = 0, 1, \ldots, L-2$, we need to carry out

$$O\left(\sum_{t=0}^{L-2} 4^{d^2\,\mathrm{mpe}(q-1)+dL-dt} 2^{-d}\right) \subseteq O(4^{d^2\,\mathrm{mpe}(q-1)+dL})$$

computations of a distribution number of $\mathcal{Q}_{i_{-t-1},H_i+L-t-2}$ of the above form, and as above (this time using that all entries of $\vec{\xi}_{i_{-t-1},H_i}$ are the positive logical sign), the bit operation cost of each individual such distribution number computation is (5.10). Therefore, we end up with a total bit operation cost of

$$O(8^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)} q)$$

for computing all functions $\mathfrak{u}_{i_{-t},L-t}$.

Next, we compute the set $\mathfrak{C}_{i,L}$, and for each $l \in \mathfrak{C}_{i,L}$, we compute the s-congruence $\eta_{i,l}(x)$. By Lemma 5.1.5 (1,3,4,8), this takes $O((d+L)\log^{1+o(1)} q)$ bit operations altogether if $\mathcal{A}_{i,l} = (A_{i_0}A_{i_1}\cdots A_{i_{\ell-1}})^{l/\ell} = \mathcal{A}_i^{l/\ell}$, with linear coefficient $\overline{\alpha}_{i,l}$ and constant coefficient $\overline{\beta}_{i,l}$, is stored for each l such that $\ell \mid l$, then computed for the next larger relevant value, $l+\ell$, using the formula $\mathcal{A}_{i,l+\ell} = \mathcal{A}_{i,l}\mathcal{A}_i$ whenever $\ell \mid l$. With these computations, we have established a spanning congruence sequence for $\mathcal{W}_{i,L}$, of length at most $d(H_i+L)+H_i+L \le d^2\,\mathrm{mpe}(q-1)+d\,\mathrm{mpe}(q-1)+dL+L$. Now, we go through the $O(2^{d(H_i+L)}L) \subseteq O(2^{d^2\,\mathrm{mpe}(q-1)+dL}L)$ logical sign tuples $\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}$ that parametrize subsets of $\mathbb{Z}/s\mathbb{Z}$ of the form $\mathcal{B}(\mathcal{W}_{i,L}, \diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l})$ – we observe that the non-empty such sets are just those blocks of $\mathcal{W}_{i,L}$ that consist of f-periodic points. For each such tuple, we compute

$$\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}) := |\mathcal{B}(\mathcal{W}_{i,L}, \diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l})|$$

as the distribution number

$$\sigma w_{i,L,\mathbf{0}}(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l}, (\emptyset, \emptyset, \dots, \emptyset)),$$

which costs

$$\begin{aligned} &O(2^{d(H_i+L)+L}(d(H_i+L)+H_i+L)\log^{1+o(1)} q) \\ &\quad \subseteq O(2^{d^2\,\mathrm{mpe}(q-1)+dL+L}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)} q) \end{aligned}$$

bit operations per cardinality to compute. If $\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}) = 0$, then we discard that case. Otherwise, we store the computed block size. Overall, this process takes

$$\begin{aligned} &O(4^{d^2\,\mathrm{mpe}(q-1)+dL}2^L(d^2L\,\mathrm{mpe}(q-1)+dL^2)\log^{1+o(1)} q) \\ &\quad \subseteq O(8^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2L\,\mathrm{mpe}(q-1)+dL^2)\log^{1+o(1)} q) \end{aligned}$$

bit operations.

We recall that in view of Lemmas 5.3.3.1 and 5.3.3.2, the blocks of $\mathcal{W}_{i,L}$ that consist of f-periodic points control the digraph isomorphism type of the connected component of Γ_f containing any given point in the block. We use Lemma 5.3.3.2 to compute, for each tuple $\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}$ whose $\mathfrak{m}_i$-value (the associated block size) is non-zero, the unique cyclic sequence $[\mathfrak{n}_1, \mathfrak{n}_2, \dots, \mathfrak{n}_{l'}]$ with entries in $\{0, 1, \dots, N\}$ and of minimal period l' such that the cyclic sequence of rooted tree isomorphism types characterizing the corresponding digraph isomorphism type is equal to

$$[\diamond_{m=1}^{l/l'}(\mathfrak{T}_{\mathfrak{n}_1}, \mathfrak{T}_{\mathfrak{n}_2}, \dots, \mathfrak{T}_{\mathfrak{n}_{l'}})].$$

To that end, for fixed $\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}$, we compute the sequence

$$(\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \dots, \mathfrak{n}^{(0)}),$$

where $\mathfrak{n}^{(-t)}$ for $t \in \{0, 1, \dots, l-1\}$ is the unique index in $\{0, 1, \dots, N\}$ such that

$$\begin{aligned} \mathrm{Tree}_{i_{-t}}(\mathcal{P}_{i_{-t}}, (\mathrm{proj}_{i_{-t},L-t} &\circ \mathfrak{u}_{i_{-t+1},L-t+1} \\ &\circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k)) \cong \mathfrak{T}_{\mathfrak{n}^{(-t)}}. \end{aligned}$$

For each fixed t, this requires us to compute the logical sign tuple

$$(\mathrm{proj}_{i_{-t},L-t} \circ \mathfrak{u}_{i_{-t+1},L-t+1} \circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k). \tag{5.11}$$

Now, we can compute

$$(\mathfrak{u}_{i_{-t+1},L-t+1} \circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k)$$

simply by looking up the pre-computed values of the functions $\mathfrak{u}_{i_{-j},L-j}$, which takes $O(t(H_i+L)d) \subseteq O(d^2 L \operatorname{mpe}(q-1) + dL^2)$ bit operations. The $\operatorname{proj}_{i_{-t},L-t}$-value of this tuple is a projection onto an initial segment, which can be read off using $O(d(H_i+1)) \subseteq O(d^2 \operatorname{mpe}(q-1))$ bit operations. Finally, we need to look up the number $\mathfrak{n}^{(-t)}$ in our partition-tree register – it is characterized by the inclusion

$$(\operatorname{proj}_{i_{-t},L-t} \circ \mathfrak{u}_{i_{-t+1},L-t+1} \circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k) \in S_{\mathfrak{n}^{(-t)},i_{-t},H_i}.$$

Therefore, in order to determine and store $\mathfrak{n}^{(-t)}$, we need to go through the pairwise disjoint sets S_{n,i_{-t},H_i} for $n = 0, 1, \ldots, N$ until we find the one that contains the logical sign tuple (5.11). Because $\sum_{n=0}^{N} |S_{n,i_{-t},H_i}| \leq 2^{d(H_i+1)} \in O(2^{d^2 \operatorname{mpe}(q-1)+d})$, and each element of each set S_{n,i_{-t},H_i} is a logical sign tuple of length in $O(d(H_i+1)) \subseteq O(d^2 \operatorname{mpe}(q-1))$, it takes

$$\begin{aligned} &O(N + 2^{d^2 \operatorname{mpe}(q-1)+d} d^2 \operatorname{mpe}(q-1) + \log q) \\ &\quad \subseteq O(d^2 \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} + \log q) \end{aligned}$$

bit operations to determine $\mathfrak{n}^{(-t)}$ for our fixed value of t. In total, the computation of the sequence $\vec{\mathfrak{n}} = (\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(0)})$ for a fixed value of $\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}$ takes

$$O(d^2 L^2 \operatorname{mpe}(q-1) + dL^3 + d^2 L \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} + L \log q)$$

bit operations. The cyclic sequence $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$ we are looking for is simply

$$[\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(-l+l')}],$$

where $l' = \operatorname{minperl}(\vec{\mathfrak{n}})$, which can be computed in

$$O(l^{1+o(1)} l_{\text{bit}}) \subseteq O(L^{1+o(1)} \log q)$$

bit operations by Lemma 5.3.2.7. Finally, in representing this cyclic sequence, we would like to replace $(\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'})$ by the lexicographically minimal ordered sequence in the same cyclic equivalence class. This takes another $O(L^2 \log L \log q)$ bit operations per such sequence. In total, the computation of the cyclic sequence

$$[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$$

for all of the $O(2^{d(H_i+L)} L) \subseteq O(2^{d^2 \operatorname{mpe}(q-1)+dL} L)$ logical sign tuples $\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}$ takes

$$\begin{aligned} &O(2^{d^2 \operatorname{mpe}(q-1)+dL} L \cdot (d^2 L^2 \operatorname{mpe}(q-1) + dL^3 + d^2 L \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} \\ &+ L^2 \log L \log q)) \subseteq O(8^{d^2 \operatorname{mpe}(q-1)+dL} \log^{1+o(1)} q) \end{aligned}$$

bit operations. In summary, the bit operation cost of the computations described after fixing $(i, \ell) \in \bar{\mathcal{L}} \setminus \{(d, 1)\}$ for all of those $O(d)$ pairs (i, ℓ) together is in

$$O(8^{d^2 \operatorname{mpe}(q-1)+dL}(d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q).$$

At this point, we have computed, for each $(i, \ell) \in \bar{\mathcal{L}} \setminus \{(d, 1)\}$, a set $\mathfrak{N}_i$ of the form

$$\begin{aligned}\mathfrak{N}_i = \{&(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}, [\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}], \mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l})) : \\ &\vec{o}'_k \in \{\emptyset, \neg\}^{n_{i-k}} \text{ for } k = 0, 1, \ldots, H_i + L - 1, l \in \mathfrak{C}_{i,L}, \\ &\text{and } \mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}) > 0\}.\end{aligned}$$

We note that for each element of $\mathfrak{N}_i$, we have $l' \mid l$ and $\operatorname{minperl}([\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]) = l'$. We also observe that while $\mathfrak{N}_i$ is *not* the tree necklace list, relative to $\vec{\mathfrak{F}}$, for the restriction of f to the union of all cosets C_j, where j is a vertex of the connected component of $\Gamma_{\bar{f}}$ containing i, that tree necklace list could be easily derived from $\mathfrak{N}_i$ as follows. If we fix l and $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$ and add up all the third entries of the corresponding triples in $\mathfrak{N}_i$ (i.e., of those triples where the first entry has terminal segment $\vec{\nu}_{i,L,l}$ and the second entry is $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$), then we end up with the exact number of connected components of Γ_f that are characterized by the cyclic sequence of rooted tree isomorphism types $[\diamond_{m=1}^{l/l'}(\mathfrak{F}_{\mathfrak{n}_1}, \ldots, \mathfrak{F}_{\mathfrak{n}_{l'}})]$ and are contained in the union of all cosets C_j for j in the connected component of $\Gamma_{\bar{f}}$ containing i. This observation also implies that we can compute the full tree necklace list $\mathfrak{N}$ of f relative to $\vec{\mathfrak{F}}$ as follows.

We start by setting $\mathfrak{N} := \emptyset$. Throughout the process described below, $\mathfrak{N}$ is a set of triples $([\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}], l, \mathfrak{m})$ such that

- $|\mathfrak{N}| \leq \sum_{(i,\ell) \in \bar{\mathcal{L}} \setminus \{(d,1)\}} |\mathfrak{N}_i| \in O(dL2^{d^2 \operatorname{mpe}(q-1)+dL})$;
- $\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'} \in \{0, 1, \ldots, N\}$;
- $\operatorname{minperl}([\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]) = l'$;
- $[\diamond_{m=1}^{l/l'}(\mathfrak{F}_{\mathfrak{n}_1}, \mathfrak{F}_{\mathfrak{n}_2}, \ldots, \mathfrak{F}_{\mathfrak{n}_{l'}})]$ is a cyclic sequence of rooted tree isomorphism types that characterizes at least one connected component of Γ_f that is contained in $\mathbb{F}_q^*$; and
- $\mathfrak{m} \in \mathbb{N}^+$ is the exact number of connected components of Γ_f that are contained in $\mathbb{F}_q^*$ and are characterized by $[\diamond_{m=1}^{l/l'}(\mathfrak{F}_{\mathfrak{n}_1}, \mathfrak{F}_{\mathfrak{n}_2}, \ldots, \mathfrak{F}_{\mathfrak{n}_{l'}})]$.

We remind the reader that the first entry of each triple in $\mathfrak{N}$ is represented by an ordered list of length exactly L (via padding) each entry of which has bit length in $O(\log q)$, and that the second and third entries of such a triple are represented by bit strings of length in $O(\log L)$ and $O(\log q)$, respectively. Now, to get an almost-final form of $\mathfrak{N}$, we loop over the pairs $(i, \ell) \in \bar{\mathcal{L}} \setminus \{(d, 1)\}$, and for each such pair, we loop over the elements of $\mathfrak{N}_i$; this double loop has $O(dL2^{d^2 \operatorname{mpe}(q-1)+dL})$ individual iterations. In each iteration, we add at most one new element to $\mathfrak{N}$, which explains

the above bound on $|\mathfrak{N}|$ that is valid throughout the process. Specifically, an iteration consists of the following steps. Associated with the triple in $\mathfrak{N}_i$ we are considering, we have the parameters l, which can be read off from the first entry $\diamond_{k=0}^{H_i+L} \vec{o}'_k \diamond \vec{v}_{i,L,l}$ of the triple using $O(l \log l) \subseteq O(L^{1+o(1)})$ bit operations (by scanning to find the first positive logical sign in $\vec{v}_{i,L,l}$ and incrementing a counter during that process), and $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$. We check whether these already occur as the first two entries of some element of $\mathfrak{N}$, which takes

$$O(dL2^{d^2 \operatorname{mpe}(q-1)+dL} \cdot L \log q) = O(dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations. If this is the case, then we end the current iteration, having added nothing to $\mathfrak{N}$. Otherwise, we compute the sum $\mathfrak{m}$ of the third entries of all triples in $\bigcup_{(j,l') \in \bar{\mathfrak{L}} \setminus \{(d,1)\}} \mathfrak{N}_j$ that have the parameters l and $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$ associated with them. This takes

$$O(dL2^{d^2 \operatorname{mpe}(q-1)+dL}(L \log L + L \log q)) = O(dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations. Then we add $([\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}], l, \mathfrak{m})$ to $\mathfrak{N}$ as a new element and end the current iteration. Overall, this double loop takes

$$\begin{aligned} &O(dL2^{d^2 \operatorname{mpe}(q-1)+dL} \cdot dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q) \\ &= O(d^2 L^3 4^{d^2 \operatorname{mpe}(q-1)+dL} \log q) \\ &\subseteq O(8^{d^2 \operatorname{mpe}(q-1)+dL} \log q) \end{aligned}$$

bit operations.

At the end of the double loop, $\mathfrak{N}$ is almost equal to the tree necklace list for f relative to $\vec{\mathfrak{F}}$; only the connected component of Γ_f containing $0_{\mathbb{F}_q}$, which is characterized by the cyclic sequence $[\operatorname{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})]$, has not yet been accounted for. To find the index number $\mathfrak{n} \in \{0, 1, \ldots, N\}$ of $\operatorname{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$ with respect to our partition-tree register, we note that $\mathfrak{n}$ is characterized by the equality $S_{\mathfrak{n},d} = \emptyset$ (the positive logical sign). Therefore, we only need an additional $O(N) \subseteq O(d2^{d^2 \operatorname{mpe}(q-1)+d})$ bit operations to find $\mathfrak{n}$. Then, we need to check whether $[\mathfrak{n}]$ and 1 already occur as the first two entries of some triple in $\mathfrak{N}$, which takes

$$O(dL2^{d^2 \operatorname{mpe}(q-1)+dL} \cdot \log q) \subseteq O(8^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations. If so, we increase the third entry of that triple by 1 and halt. Otherwise, we add $([\mathfrak{n}], 1, 1)$ to $\mathfrak{N}$ as a new element.

Finally, we sort the computed array representing $\mathfrak{N}$ lexicographically. By Lemma 5.1.5 (10), since $|\mathfrak{N}| \in O(dL2^{d^2 \operatorname{mpe}(q-1)+dL})$ and each entry of the array has bit length in $O(L \log q)$, this takes

$$O(dL^2(d^2 \operatorname{mpe}(q-1) + dL)2^{d^2 \operatorname{mpe}(q-1)+dL} \log q) \subseteq O(8^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations. We conclude by outputting $(\vec{\mathfrak{D}}, \mathfrak{N})$ and halting. ∎

Of course, Theorem 5.3.3.4 is not useful in practice unless the maximum cycle length of a generalized cyclotomic mapping can be computed efficiently. The following proposition takes care of that.

Proposition 5.3.3.5. *Let f be an index d generalized cyclotomic mapping of $\mathbb{F}_q$. The maximum cycle length of $f_{|\mathrm{per}(f)}$ can be computed within q-bounded query complexity*

$$(d \log^{2+o(1)} q + d \log^2 d, d, 1, d, 0).$$

Proof. This is similar in spirit to Proposition 5.3.2.3, but we obtain a better bound than by using that proposition directly, because there is no need to spell out entire cycle types here. First, we compute $\bar{f}$, the affine maps A_i, the cycles of $\bar{f}$ and a CRL-list $\bar{\mathcal{L}}$ of $\bar{f}$. By Proposition 5.1.8 and the beginning of Section 5.2.1, this takes q-bounded query complexity

$$(d \log^{1+o(1)} q + d \log^2 d, d, 0, 0, 0).$$

We also factor s, using a single q-bounded mdl query (i.e., spending q-bounded query complexity $(\log q, 0, 1, 0, 0)$).

Now, we loop over the pairs $(i, \ell) \in \bar{\mathcal{L}} \setminus \{(d, 1)\}$, and for each of them, we do the following. First, we compute $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}} : x \mapsto \bar{\alpha}_i x + \bar{\beta}_i$, taking $O(\ell \log^{1+o(1)} q)$ bit operations (per (i, ℓ)) by the beginning of Section 5.2.1. Next, we determine the largest cycle length of $\mathcal{A}_i$ on $\mathrm{per}(\mathcal{A}_i) \subseteq \mathbb{Z}/s\mathbb{Z}$. To do so, we note that every cycle length of $\mathcal{A}_i$ is a least common multiple of cycle lengths of those primary components $\bar{\mathcal{A}}_{i,p} = \mathcal{A}_i \bmod p^{\nu_p(s)}$, where $p \mid s$ and $p \nmid \bar{\alpha}_i$. But by [15, Tables 3 and 4] (or our Table 2.2), the largest cycle length of $\bar{\mathcal{A}}_{i,p}$ is equal to the order of $\bar{\mathcal{A}}_{i,p}$ in $\mathrm{Sym}(\mathbb{Z}/p^{\nu_p(s)}\mathbb{Z})$, i.e., to the least common multiple of all cycle lengths of $\bar{\mathcal{A}}_{i,p}$. It follows that the largest cycle length of $\mathcal{A}_i$ is equal to $\mathrm{ord}(\mathcal{A}_i \bmod s_i')$, where $s_i' = \prod_{p \mid s, p \nmid \bar{\alpha}_i} p^{\nu_p(s)}$. We compute this order as follows.

We set $s_i' := 1$ and loop over the $O(\log q)$ primes p dividing s (which can be read off from the factorization of s computed above). For each p, we check whether $p \mid \bar{\alpha}_i$, taking $O(\log^{1+o(1)} q)$ bit operations by Lemma 5.1.5 (3). If so, we skip to the next p; otherwise, we overwrite $s_i' := s_i' \cdot p^{\nu_p(s)}$, taking $O(\log^{1+o(1)} q)$ bit operations for the multiplication (there is no need to compute the power $p^{\nu_p(s)}$, because it is part of the output of the mdl query used to factor s). At the end of this loop over p, which has a bit operation cost in $O(\log^{2+o(1)} q)$, the variable s_i' has the desired value. Now, we compute $\bar{\alpha}_i' := \bar{\alpha}_i \bmod s_i'$ and $\bar{\beta}_i' := \bar{\beta}_i \bmod s_i'$, taking $O(\log^{1+o(1)} q)$ bit operations, to get the affine map $\mathcal{A}_i' = \mathcal{A}_i \bmod s_i' : x \mapsto \bar{\alpha}_i' x + \bar{\beta}_i'$ of $\mathbb{Z}/s_i'\mathbb{Z}$. We wish to compute $\mathrm{ord}(\mathcal{A}_i')$. To that end, we first compute the (multiplicative) order $\mathrm{ord}_{s_i'}(\bar{\alpha}_i')$ with a q-bounded mord query. We observe that $\mathrm{ord}_{s_i'}(\bar{\alpha}_i')$ divides $\mathrm{ord}(\mathcal{A}_i')$, because $(\mathcal{A}_i')^t(x) = (\bar{\alpha}_i')^t x + c(\bar{\alpha}_i', \bar{\beta}_i', t)$ for all $x \in \mathbb{Z}/s_i'\mathbb{Z}$ and all $t \in \mathbb{Z}$. Therefore,

$$\mathrm{ord}(\mathcal{A}_i') = \mathrm{ord}_{s_i'}(\bar{\alpha}_i') \cdot \mathrm{ord}((\mathcal{A}_i')^{\mathrm{ord}_{s_i'}(\bar{\alpha}_i')}).$$

But $(\mathcal{A}'_i)^{\mathrm{ord}_{s'_i}(\overline{\alpha}'_i)}$ is the translation $x \mapsto x + \overline{\beta}''_i$ such that

$$\overline{\beta}''_i = \begin{cases} \overline{\beta}'_i, & \text{if } \overline{\alpha}'_i = 1, \\ \overline{\beta}'_i \frac{(\overline{\alpha}'_i)^{\mathrm{ord}_{s'_i}(\overline{\alpha}'_i)} - 1}{\overline{\alpha}'_i - 1}, & \text{otherwise,} \end{cases}$$

where the formula in the second case is to be evaluated in the ring $\mathbb{Z}$, although the result is to be viewed as an element of $\mathbb{Z}/s'_i\mathbb{Z}$. Because $\mathrm{ord}((\mathcal{A}'_i)^{\mathrm{ord}_{s'_i}(\overline{\alpha}'_i)})$ is equal to the additive order of $\overline{\beta}''_i$ modulo s'_i, we can work out $\mathrm{ord}(\mathcal{A}'_i)$, which coincides with the largest cycle length of $\mathcal{A}_i$, using another $O(\log^{2+o(1)} q) = O(\log^{2+o(1)}(q^2))$ bit operations (for computing $\overline{\beta}''_i \in \mathbb{Z}/s'_i\mathbb{Z}$, which may involve a power computation modulo $s'_i(\overline{\alpha}'_i - 1) \in O(q^2)$, and working out its additive order modulo s'_i via a gcd computation and a division). We conclude this loop by setting $l_i := \ell \cdot \mathrm{ord}(\mathcal{A}'_i)$, which takes $O(\log^{1+o(1)} q)$ bit operations to compute and is the largest cycle length of f on its periodic points in $\bigcup_{t=0}^{\ell-1} C_{i_t}$. This ends our description of the loop over $(i, \ell) \in \overline{\mathcal{L}} \setminus \{(d, 1)\}$, which overall takes q-bounded query complexity $(d \log^{2+o(1)} q, 0, 0, d, 0)$, using that $\sum_{(i,\ell)\in\overline{\mathcal{L}}} \ell \leq d$.

Finally, the maximum cycle length of f is simply the maximum value among the l_i for $(i, \ell) \in \overline{\mathcal{L}}$, where $l_d := 1$, which takes $O(d \log q)$ bit operations to compute. ∎

To conclude, we give the following corollary of Theorem 5.3.3.4, which can be seen as the main result of this subsection.

Corollary 5.3.3.6. *Let f_1 and f_2 be generalized cyclotomic mappings of $\mathbb{F}_q$, of index d_1 and d_2, respectively, and set $d := \max\{d_1, d_2\}$. Moreover, let $L \in \mathbb{N}^+$, and denote by L_1, respectively, L_2, the maximum cycle length of f_1, respectively of f_2, on its periodic points. Then, if $\min\{L_1, L_2\} \leq L$, it can be decided whether $\Gamma_{f_1} \cong \Gamma_{f_2}$ within q-bounded query complexity*

$$\begin{aligned} &(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{d \operatorname{mpe}(q-1)} (d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q \\ &\quad + d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log^2 q + d \log^{2+o(1)} q, \\ &\quad d, 1, d, 0), \end{aligned}$$

and thus within q-bounded Las Vegas dual complexity

$$\begin{aligned} &(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{d \operatorname{mpe}(q-1)} (d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q \\ &\quad + d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log^2 q + d \log^{2+o(1)} q + d \log^{7+o(1)} q, \\ &\quad d \log^{3+o(1)} q, d \log q). \end{aligned}$$

Proof. First, we compute L_1 and L_2, which takes q-bounded query complexity

$$(d \log^{2+o(1)} q + d \log^2 d, d, 1, d, 0)$$

by Proposition 5.3.3.5. We check whether $L_1 = L_2$, taking $O(\log q)$ bit operations. If not, then $\Gamma_{f_1} \ncong \Gamma_{f_2}$, so we may output "false" and halt. Otherwise, we continue by computing, for $j = 1, 2$,

- a recursive tree description list $\vec{\mathfrak{D}}^{(j)} = (\mathfrak{D}_n^{(j)})_{n=0,1,\ldots,N_j}$, with

$$N_j \in O(d2^{d^2 \operatorname{mpe}(q-1)+d})$$

  and associated rooted tree isomorphism type list $\vec{\mathfrak{F}}^{(j)}$; and
- the tree necklace list $\mathfrak{N}_j$ of f_j relative to $\vec{\mathfrak{F}}^{(j)}$ such that

$$|\mathfrak{N}_j| \in O(dL2^{d^2 \operatorname{mpe}(q-1)+dL}).$$

By Theorem 5.3.3.4, this can be done within q-bounded query complexity

$$(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{d \operatorname{mpe}(q-1)} (d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q, d, 0, 0, 0).$$

Next, we compute a synchronization $(\vec{\mathfrak{D}}^+, \mathrm{i})$ of $\vec{\mathfrak{D}}^{(1)}$ and $\vec{\mathfrak{D}}^{(2)}$, in the sense of Definition 5.3.2.11. By Lemma 5.3.2.12, this takes

$$\begin{aligned} &O((d^3 8^{d^2 \operatorname{mpe}(q-1)+d} + d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log q) \cdot \log q) \\ &= O(d^3 8^{d^2 \operatorname{mpe}(q-1)+d} \log q + d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log^2 q) \end{aligned}$$

bit operations. Following that, we overwrite each first entry $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$ in each triple in $\mathfrak{N}_2$ with $[\mathrm{i}(\mathfrak{n}_1), \mathrm{i}(\mathfrak{n}_2), \ldots, \mathrm{i}(\mathfrak{n}_{l'})]$, using lexicographically minimal representatives of cyclic equivalence classes, which results in a modified, unsorted tree necklace list $\mathfrak{N}_2'$. This takes

$$O(dL2^{d^2 \operatorname{mpe}(q-1)+dL} \cdot L^2 \log L \log q) = O(dL^3 \log L 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations overall. After this, we sort $\mathfrak{N}_2'$ lexicographically, which takes

$$O(dL^2(d^2 \operatorname{mpe}(q-1) + dL)2^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations (see also the end of the proof of Theorem 5.3.3.4). Finally, we note that $\Gamma_{f_1} \cong \Gamma_{f_2}$ if and only if $\mathfrak{N}_1 = \mathfrak{N}_2'$, so we determine the truth value of the latter, which takes

$$O(dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q)$$

bit operations using a linear scan. We then output that truth value and halt. ∎

Like the previous subsections, we conclude this subsection with some pseudocode for the discussed algorithms, specifying the q-bounded query complexity (QC) of each step. We begin with the algorithm from Theorem 5.3.3.4, which on input (f, L), where f is an index d generalized cyclotomic mapping f of $\mathbb{F}_q$ such that all cycle

lengths of f are at most L, computes a pair $(\vec{\mathfrak{D}}, \mathfrak{N})$ such that $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$ is a recursive tree description list with $N \in O(d2^{d^2 \operatorname{mpe}(q-1)+d})$ and associated sequence of rooted tree isomorphism types $\vec{\mathfrak{T}}$, and $\mathfrak{N}$ is a tree necklace list for f relative to $\vec{\mathfrak{T}}$ with $|\mathfrak{N}| \in O(dL2^{d^2 \operatorname{mpe}(q-1)+dL+d})$.

1 Compute the induced function $\bar{f}$ on $\{0, 1, \ldots, d\}$ and the affine maps A_i of $\mathbb{Z}/s\mathbb{Z}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Compute a partition-tree register

$$((\mathcal{Z}_i)_{i=0,1,\ldots,d-1}, ((\mathfrak{D}_n, (S_{n,i})_{i=0,1,\ldots,d}))_{n=0,1,\ldots,N})$$

of f with $N \in O(d2^{d^2 \operatorname{mpe}(q-1)+d})$. In the process, store a CRL-list $\bar{\mathfrak{L}}$ of $\bar{f}$, the cycles of $\bar{f}$, and the parameter H_i for each $\bar{f}$-periodic $i \in \{0, 1, \ldots, d-1\}$.
QC: $(d^3 \operatorname{mpe}(q-1)2^{(3d^2+d)\operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q, 0, 0, 0, 0)$ because $\bar{f}$ and the A_i have already been computed.

3 Set $\vec{\mathfrak{D}} := (\mathfrak{D}_n)_{n=0,1,\ldots,N}$.
QC: $(d^2 4^{d^2 \operatorname{mpe}(q-1)+d} \log q, 0, 0, 0, 0)$.

4 For each $(i, \ell) \in \bar{\mathfrak{L}} \setminus \{(d, 1)\}$, do the following.
QC: $(8^{d^2 \operatorname{mpe}(q-1)+dL}(d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1 For each $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(d^3 \operatorname{mpe}(q-1) \log q + d^2 L \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.1 From $\mathcal{Z}_{i_t}$, read off a spanning congruence sequence for $\mathcal{U}_{i_t}$ of length $H_i \le d \operatorname{mpe}(q-1)$.
QC: $(d \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

4.1.2 For each $j = 0, 1, \ldots, H_i + L - 1$, do the following.
QC: $(d^2 \operatorname{mpe}(q-1) \log q + dL \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.2.1 If $j \le H_i$, then do the following.

4.1.2.1.1 From $\mathcal{Z}_{i_{t+j}}$, read off a spanning congruence sequence for $\lambda_{i_t}^j(\mathcal{R}_{i_t})$ of length $n_{i_t} \le d$.
QC: $(d \log q, 0, 0, 0, 0)$.

4.1.2.2 Else do the following.

4.1.2.2.1 Compute a spanning congruence sequence for

$$\lambda_{i_t}^j(\mathcal{R}_{i_t}) = \lambda(\lambda_{i_t}^{j-1}(\mathcal{R}_{i_t}), A_{i_{t+j-1}})$$

of length $n_{i_t} \le d$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.2 For each $t = 0, 1, \ldots, L - 1$, do the following.
QC: $((d^2 L \operatorname{mpe}(q-1) + dL^2) \log q, 0, 0, 0, 0)$.

4.2.1 From the data stored in step 4.1, paste together a spanning congruence sequence for

$$\mathcal{Q}_{i_{-t},H_i+L-t-1} = \bigwedge_{j=0}^{H_i+L-t-1} \lambda^j_{i_{-t-j}}(\mathcal{R}_{i_{-t-j}}) \wedge \mathcal{U}_{i_{-t}}.$$

QC: $((d^2 \operatorname{mpe}(q-1) + dL) \log q, 0, 0, 0, 0)$.

4.3 For each $t = 0, 1, \ldots, L-1$, do the following.
QC: $(4^{d^2 \operatorname{mpe}(q-1)+dL}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.3.1 Set $\mathcal{O}_{i_{-t},L-t} := \emptyset$.
QC: $(\log d + \log L, 0, 0, 0, 0)$.

4.3.2 For each $\diamond_{k=0}^{H_i+L-t-1} \vec{o'_k} \in \{\emptyset, \neg\}^{n_{i_{-t}}+n_{i_{-t-1}}+\cdots+n_{i_{-H_i-L+1}}}$, do the following.
QC: $(4^{d^2 \operatorname{mpe}(q-1)+dL} 2^{-dt}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.3.2.1 Check whether

$$\sigma_{\mathcal{Q}_{i_{-t},H_i+L-t-1},\mathbf{0}}(\diamond_{k=0}^{H_i+L-t-1} \vec{o'_k} \diamond \vec{\xi}_{i_{-t},H_i}, (\emptyset, \ldots, \emptyset)) > 0,$$

and if so, add $\diamond_{k=0}^{H_i+L-t-1} \vec{o'_k}$ to $\mathcal{O}_{i_{-t},L-t}$ as a new element.
QC: $(2^{d^2 \operatorname{mpe}(q-1)+dL}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.4 For each $t = 0, 1, \ldots, L-2$, do the following.
QC: $(8^{d^2 \operatorname{mpe}(q-1)+dL}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.4.1 For each $\diamond_{k=0}^{H_i+L-t-1} \vec{o'_k} \in \mathcal{O}_{i_{-t},L-t}$, do the following.
QC: $(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{-d(t+1)} 2^{-dt}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q$, $0, 0, 0, 0)$.

4.4.1.1 For each $\diamond_{k=0}^{H_i+L-t-2} \vec{o_k} \in \mathcal{O}_{i_{-t-1},L-t-1}$, do the following.
QC: $(4^{d^2 \operatorname{mpe}(q-1)+dL} 2^{-d(t+1)}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q$, $0, 0, 0, 0)$.

4.4.1.1.1 Check whether the distribution number

$$\sigma_{\mathcal{Q}_{i_{-t-1},H_i+L-t-2},A_{i_{-t-1}}}(\diamond_{k=0}^{H_i+L-t-2} \vec{o_k} \diamond \vec{\xi}_{i_{-t-1},H_i}, \diamond_{k=1}^{H_i+L-t-1} \vec{o'_k} \diamond \vec{\xi}_{i_{-t},H_i})$$

is equal to 1. If so, set

$$\mathfrak{u}_{i_{-t},L-t}(\diamond_{k=0}^{H_i+L-t-1} \vec{o'_k}) := \diamond_{k=0}^{H_i+L-t-2} \vec{o_k}.$$

QC: $(2^{d^2 \operatorname{mpe}(q-1)+dL}(d^2 \operatorname{mpe}(q-1)+dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.5 Compute $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.6 Set $\mathfrak{C}_{i,L} := \emptyset$.
QC: $(\log d + \log L, 0, 0, 0, 0)$.

4.7 For each $l \in \{1, 2, \ldots, L\}$, do the following.
QC: $(L \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.7.1 Check whether $\ell \mid l$. If not, skip to the next l.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.7.2 Compute and store the affine iterate

$$\mathcal{A}_{i,l} = \mathcal{A}_i^{l/\ell} = \begin{cases} \mathcal{A}_i, & \text{if } l = \ell, \\ \mathcal{A}_{i,l-\ell} \mathcal{A}_i, & \text{if } \ell \mid l > \ell, \end{cases}$$

with linear coefficient $\overline{\alpha}_{i,l}$ and constant coefficient $\overline{\beta}_{i,l}$.
QC: $(\log q, 0, 0, 0, 0)$ if $l = \ell$ (only needing to copy information from step 4.5); $(\log^{1+o(1)} q, 0, 0, 0, 0)$ otherwise.

4.7.3 Check whether $\gcd(s, \overline{\alpha}_{i,l} - 1) \mid \overline{\beta}_{i,l}$. If so, add l to $\mathfrak{C}_{i,L}$ as a new element, and store $\eta_{i,l}(x)$ as the s-congruence

$$x \equiv -\frac{\overline{\beta}_{i,l}}{\gcd(s, \overline{\alpha}_{i,l} - 1)} \cdot \operatorname{inv}_{\frac{s}{\gcd(s, \overline{\alpha}_{i,l}-1)}} \left(\frac{\overline{\alpha}_{i,l} - 1}{\gcd(s, \overline{\alpha}_{i,l} - 1)} \right) \cdot \left(\operatorname{mod} \frac{s}{\gcd(s, \overline{\alpha}_{i,l} - 1)} \right).$$

QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.8 Set $\mathfrak{N}_i := \emptyset$.
QC: $(\log d, 0, 0, 0, 0)$.

4.9 For each $\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \in \mathcal{O}_{i,L}$, do the following.
QC: $(4^{d^2 \operatorname{mpe}(q-1)+dL} 2^L (d^2 L \operatorname{mpe}(q-1) + dL^2) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.9.1 For each $l \in \mathfrak{C}_{i,L}$, do the following.
QC: $(2^{d^2 \operatorname{mpe}(q-1)+dL+L} (d^2 L \operatorname{mpe}(q-1) + dL^2) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.9.1.1 Compute the logical sign tuple $\vec{\nu}_{i,L,l}$ (see the paragraph before Proposition 5.3.3.3).
QC: $(L \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.9.1.2 Set

$$\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}) := \sigma \mathfrak{w}_{i,L,\mathbf{0}}(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l}, (\emptyset, \ldots, \emptyset)).$$

QC: $(2^{d^2 \operatorname{mpe}(q-1)+dL+L} (d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.9.1.3 If $\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}) = 0$, then skip to the next l.
QC: $(d^2 \operatorname{mpe}(q-1) + dL, 0, 0, 0, 0)$.

4.9.1.4 For each $t = 0, 1, \ldots, l-1$, do the following.
QC: $(dL^3 + d^2 L \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} + L \log q, 0, 0, 0, 0)$.

4.9.1.4.1 Compute the logical sign tuple

$$(\operatorname{proj}_{i_{-t},L-t} \circ \mathfrak{u}_{i_{-t+1},L-t+1} \circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k).$$

QC: $(d^2 L \operatorname{mpe}(q-1) + dL^2, 0, 0, 0, 0)$.

4.9.1.4.2 For each $n = 0, 1, \ldots, N$, do the following.
QC: $(d^2 \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} + \log q, 0, 0, 0, 0)$.

4.9.1.4.2.1 If the logical sign tuple computed in step 4.9.1.4.1 is an element of S_{n,i_{-t},H_i} (the last entry of the tuple $S_{n,i_{-t}}$ from the partition-tree register computed in step 2), then set $\mathfrak{n}^{(-t)} := n$ and skip to the next t.
QC: $(\max\{1, |S_{n,i_{-t},H_i}| d^2 \operatorname{mpe}(q-1)\}, 0, 0, 0, 0)$ if the condition is *not* satisfied; $(\max\{1, |S_{n,i_{-t},H_i}| d^2 \operatorname{mpe}(q-1)\} + \log q, 0, 0, 0, 0)$ if the condition *is* satisfied (the additional $O(\log q)$ bit operations are from copying the value of n; we do not need to process the $O(\log q)$-bit indices n during the loop over them, because we may jump to a neighboring address in memory in $O(1)$ bit operations).

4.9.1.5 Set $\vec{\mathfrak{n}} := (\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(0)})$.
QC: $(L \log q, 0, 0, 0, 0)$.

4.9.1.6 Set l' to be $\operatorname{minperl}(\vec{\mathfrak{n}})$.
QC: $(L^{1+o(1)} \log q, 0, 0, 0, 0)$.

4.9.1.7 Set $\vec{\mathfrak{n}}'$ to be the lexicographically minimal ordered sequence in the same cyclic equivalence class as $(\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(-l+l')})$.
QC: $(L^2 \log L \log q, 0, 0, 0, 0)$.

4.9.1.8 Add

$$(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}, [\vec{\mathfrak{n}}'], \mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1}\vec{o}'_k \diamond \vec{\nu}_{i,L,l}))$$

to $\mathfrak{N}_i$ as a new element.
QC: $(d^2 \operatorname{mpe}(q-1) + dL + L \log q, 0, 0, 0, 0)$.

5 Set $\mathfrak{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

6 For each $(i, \ell) \in \overline{\mathfrak{L}} \setminus \{(d, 1)\}$, do the following.
QC: $(d^4L^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+dL} + d^2L^2 4^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

6.1 For each $(\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}, [\vec{\mathfrak{n}}], \mathfrak{m}') \in \mathfrak{N}_i$, do the following.
QC: $(d^3L^4 \operatorname{mpe}(q-1)4^{d^2 \operatorname{mpe}(q-1)+dL} + dL^2 4^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

6.1.1 From the first entry, $\diamond_{k=0}^{H_i+L-1} \vec{o}'_k \diamond \vec{\nu}_{i,L,l}$, determine the binary representation of l.
QC: $(L \log L, 0, 0, 0, 0)$.

6.1.2 Check whether $[\vec{\mathfrak{n}}]$ and l already occur as the first two entries of some element of $\mathfrak{N}$. If so, skip to the next element of $\mathfrak{N}_i$.
QC: $(dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

6.1.3 Compute the sum $\mathfrak{m}$ of the third entries of all triples in $\bigcup_{(j,\ell') \in \overline{\mathfrak{L}} \setminus \{(d,1)\}} \mathfrak{N}_j$ that have the parameters l and $[\vec{\mathfrak{n}}]$ associated with them.
QC: $(dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

6.1.4 Add $([\vec{\mathfrak{n}}], l, \mathfrak{m})$ to $\mathfrak{N}$ as a new element.
QC: $(L \log q, 0, 0, 0, 0)$.

7 For each $n \in \{0, 1, \ldots, N\}$, do the following.
QC: $(d 2^{d^2 \operatorname{mpe}(q-1)+d} + \log q, 0, 0, 0, 0)$.

7.1 Check whether $S_{n,d} = \emptyset$. If so, set $\mathfrak{n} := n$ and exit the loop.
QC: $(\log q, 0, 0, 0, 0)$ if the condition is satisfied (which happens only once), $(1, 0, 0, 0, 0)$ otherwise.

8 Check whether $[\mathfrak{n}]$ and 1 already occur as the first two entries of some (unique) triple in $\mathfrak{N}$, and store this information (the truth value and, if applicable, the position of that triple in $\mathfrak{N}$).
QC: $(dL 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

9 If $[\mathfrak{n}]$ and 1 occur as the first two entries of a triple $([\mathfrak{n}], 1, \mathfrak{m})$ in $\mathfrak{N}$, then do the following.

9.1 Overwrite the entry $([\mathfrak{n}], 1, \mathfrak{m})$ of the list $\mathfrak{N}$ with $([\mathfrak{n}], 1, \mathfrak{m} + 1)$.
QC: $(\log q, 0, 0, 0, 0)$.

10 Else do the following.

10.1 Add $([\mathfrak{n}], 1, 1)$ to $\mathfrak{N}$ as a new element.
QC: $(L \log q, 0, 0, 0, 0)$.

11 Sort $\mathfrak{N}$ lexicographically.
QC: $(dL^2(d^2 \operatorname{mpe}(q-1) + dL) 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

12 Output $(\vec{\mathfrak{D}}, \mathfrak{N})$ and halt.
QC: $((d^2 4^{d^2 \operatorname{mpe}(q-1)+d} + dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL}) \log q, 0, 0, 0, 0)$.

Next, we give pseudocode for the algorithm from Proposition 5.3.3.5, which for a given index d generalized cyclotomic mapping f of $\mathbb{F}_q$ outputs the maximum cycle length of f.

1 Compute the induced function $\bar{f}$ on $\{0, 1, \ldots, d\}$ and the affine maps A_i of $\mathbb{Z}/s\mathbb{Z}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Compute a CRL-list $\bar{\mathcal{L}}$ of $\bar{f}$ and the cycles of $\bar{f}$.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

3 Factor $s = p_1^{v_1} p_2^{v_2} \cdots p_K^{v_K}$.
QC: $(\log q, 0, 1, 0, 0)$.

4 For each $(i, \ell) \in \bar{\mathcal{L}} \setminus \{(d, 1)\}$, do the following.
QC: $(d \log^{2+o(1)} q, 0, 0, d, 0)$.

4.1 Compute $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}} : x \mapsto \bar{\alpha}_i x + \bar{\beta}_i$.
QC: $(\ell \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.2 Set $s_i' := 1$.
QC: $(\log d, 0, 0, 0, 0)$.

4.3 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

4.3.1 Check whether $p_j \mid \bar{\alpha}_i$, and if so, skip to the next j.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.3.2 Set $s_i' := s_i' \cdot p_j^{v_j}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.4 Set $\bar{\alpha}_i' := \bar{\alpha}_i \bmod s_i'$ and $\bar{\beta}_i' := \bar{\beta}_i \bmod s_i'$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.5 Compute $\operatorname{ord}_{s_i'}(\bar{\alpha}_i')$.
QC: $(\log q, 0, 0, 1, 0)$.

4.6 Compute

$$\bar{\beta}_i'' := \begin{cases} \bar{\beta}_i', & \text{if } \bar{\alpha}_i' = 1, \\ \bar{\beta}_i' \frac{(\bar{\alpha}_i')^{\operatorname{ord}_{s_i'}(\bar{\alpha}_i')} - 1}{\bar{\alpha}_i' - 1}, & \text{otherwise.} \end{cases}$$

In the second case, do *not* compute $(\bar{\alpha}_i')^{\operatorname{ord}_{s_i'}(\bar{\alpha}_i')}$ as an integer, but compute its value modulo $s_i'(\bar{\alpha}_i' - 1)$. Then the value modulo s_i' of the entire fraction can be determined through integer division.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

4.7 Compute the additive order of $\bar{\beta}_i''$ modulo s_i', which is equal to $s_i' / \gcd(s_i', \bar{\beta}_i'')$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.8 Set $l_i := \ell \cdot \mathrm{ord}_{s_i'}(\overline{\alpha}_i') \cdot s_i' / \gcd(s_i', \overline{\beta}_i'')$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

5 Set $L := 1$.
QC: $(\log d, 0, 0, 0, 0)$.

6 For each $(i, \ell) \in \overline{\mathfrak{L}} \setminus \{(d, 1)\}$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

6.1 If $l_i > L$, then set $L := l_i$.
QC: $(\log q, 0, 0, 0, 0)$.

7 Output L and halt.
QC: $(\log q, 0, 0, 0, 0)$.

Finally, we give pseudocode for a variant of the algorithm from Corollary 5.3.3.6. On input (L, f_1, f_2), where $L \in \mathbb{N}^+$ and f_j, for $j = 1, 2$, is a generalized cyclotomic mapping of $\mathbb{F}_q$ of index d_j, this algorithm outputs "fail" if neither the largest cycle length of f_1 nor the largest cycle length of f_2 is at most L. Otherwise, it outputs the truth value of the digraph isomorphism relation $\Gamma_{f_1} \cong \Gamma_{f_2}$.

1 For $j = 1, 2$, compute the largest cycle length L_j of f_j.
QC: $(d \log^{2+o(1)} q + d \log^2 d, d, 1, d, 0)$.

2 Check whether $\min\{L_1, L_2\} \le L$, and store this information.
QC: $(\log q, 0, 0, 0, 0)$.

3 If $\min\{L_1, L_2\} > L$, then do the following.

3.1 Output "fail" and halt.
QC: $(1, 0, 0, 0, 0)$.

4 Else do the following.

4.1 Check whether $L_1 = L_2$. If not, output "false" and halt.
QC: $(\log q, 0, 0, 0, 0)$.

5 For $j = 1, 2$, compute

- a recursive tree description list $\vec{\mathfrak{D}}^{(j)} = (\mathfrak{D}_n^{(j)})_{n=0,1,\ldots,N_j}$ with

$$N_j \in O(d 2^{d^2 \operatorname{mpe}(q-1)+d})$$

and associated rooted tree isomorphism type sequence $\vec{\mathfrak{F}}^{(j)}$; and

- the tree necklace list $\mathfrak{N}_j$ of f_j relative to $\vec{\mathfrak{F}}^{(j)}$, with

$$|\mathfrak{N}_j| \le O(dL 2^{d^2 \operatorname{mpe}(q-1)+dL}).$$

QC: $(8^{d^2 \operatorname{mpe}(q-1)+dL} 2^{d \operatorname{mpe}(q-1)} (d^3 L \operatorname{mpe}(q-1) + d^2 L^2) \log^{1+o(1)} q, d, 0, 0, 0)$.

6 Compute a synchronization $(\vec{\mathfrak{D}}^+, \mathrm{i})$ of $\vec{\mathfrak{D}}^{(1)}$ and $\vec{\mathfrak{D}}^{(2)}$.
QC: $(d^3 8^{d^2 \operatorname{mpe}(q-1)+d} \log q + d^2 4^{d^2 \operatorname{mpe}(q-1)} \log^2 q, 0, 0, 0, 0)$.

7 Set $\mathfrak{N}_2' := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

8 For each $([\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}], l, \mathfrak{m}) \in \mathfrak{N}_2$, do the following.
QC: $(dL^3 \log L 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

8.1 Compute the lexicographically minimal representative of the cyclic equivalence class $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$, representing that class by it.
QC: $(L^2 \log L \log q, 0, 0, 0, 0)$.

8.2 Add $([\mathfrak{i}(\mathfrak{n}_1), \mathfrak{i}(\mathfrak{n}_2), \ldots, \mathfrak{i}(\mathfrak{n}_{l'})], l, \mathfrak{m})$ to $\mathfrak{N}_2'$ as a new element.
QC: $(L \log q, 0, 0, 0, 0)$.

9 Sort $\mathfrak{N}_2'$ lexicographically.
QC: $(dL^2(d^2 \operatorname{mpe}(q-1) + dL)2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

10 Check whether $\mathfrak{N}_1 = \mathfrak{N}_2'$ as sets, output the truth value of this equality, and halt.
QC: $(dL^2 2^{d^2 \operatorname{mpe}(q-1)+dL} \log q, 0, 0, 0, 0)$.

Chapter 6

Open problems

We conclude this memoir with a discussion of open problems related to our results and methods.

6.1 Asymptotic behavior of mpe over prime powers

Our Proposition 5.1.10 states that as q ranges over an initial segment of all prime powers, the average value of $\mathrm{mpe}(q-1)$ (the maximum exponent of a prime in the full factorization of $q-1$) is bounded from above by a constant (independent of that segment). This led to the important observation that when d is fixed, then for asymptotically almost all finite fields $\mathbb{F}_q$, the complexities in Theorem 5.1.9 (2,3) are polynomial in $\log q$. Following that, at the end of Section 5.1, we raised the analogous problem restricted to powers of 2, and our computational evidence (gathered in the form of Table 5.1) leads to the following conjecture.

Conjecture 6.1.1. *The average value of* $\mathrm{mpe}(2^k-1)$*, where* k *ranges over an initial segment of* $\mathbb{N}^+$*, is always less than* 2*. Formally, this conjecture asserts that for each* $K \in \mathbb{N}^+$*, one has*

$$\frac{1}{K}\sum_{k=1}^{K} \mathrm{mpe}(2^k-1) < 2.$$

In fact, looking at Table 5.1, one might even conjecture that the said average value is always less than $3/2$, which would imply that $\mathrm{mpe}(2^k-1)=1$ (i.e., that 2^k-1 is square-free) for more than half of all $k \in \mathbb{N}^+$. We do note that the conjecture already fails when the prime 2 is replaced by 3. Indeed, since $4 = 2^2 \mid 3^k-1$ whenever $2 \mid k$, and $16 = 2^4 \mid 3^k-1$ whenever $4 \mid k$, one has

$$\frac{1}{K}\sum_{k=1}^{K} \mathrm{mpe}(3^k-1) \geq \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{4}\cdot 4 = 2$$

whenever $4 \mid K$. In general, there cannot be a universal constant c such that

$$\frac{1}{K}\sum_{k=1}^{K} \mathrm{mpe}(p^k-1) < c,$$

since $\mathrm{mpe}(p^k-1) \geq \mathrm{mpe}(p-1)$ for all k, and $\mathrm{mpe}(p-1)$ can be arbitrarily large (as follows, e.g., from Dirichlet's theorem on primes in arithmetic progressions, see [8, Chapter 7]). Of course, as a more general conjecture, it may be possible that for each

prime p, there is a constant c_p such that $\frac{1}{K}\sum_{k=1}^{K} \mathrm{mpe}(p^k - 1) < c_p$ for all positive integers K (and Conjecture 6.1.1 simply asserts that $c_2 = 2$ is a valid choice).

6.2 Efficient comparison of arithmetic partitions

We recall Problem 3 from the beginning of Chapter 5: given a generalized cyclotomic mapping f of $\mathbb{F}_q$ and a pair (r, l), where $r \in \mathbb{F}_q$ is f-periodic of cycle length l, the task is to obtain a compact description of the digraph isomorphism type of the connected component of Γ_f containing r, viewed as a necklace of rooted tree isomorphism types. In Problem 3, it is also assumed that a partition-tree register for f (in the sense of Definition 5.1.2) is given, and we can use this to refer to the rooted trees with their numbers $n \in \{0, 1, \ldots, N\}$ in this register, rather than spell each of them out completely.

Assuming that $r \neq 0_{\mathbb{F}_q}$ (the case "$r = 0_{\mathbb{F}_q}$" is easily dealt with separately), the main idea behind our algorithm from the proof of Theorem 5.1.9 (3) for tackling this problem is to identify the positions on the f-cycle of r that lie in a given coset C_j with the elements of $\mathbb{Z}/(l/\ell)\mathbb{Z}$, where ℓ is the associated coset cycle length (i.e., the $\bar{f}$-cycle length of i for the unique $i \in \{0, 1, \ldots, d-1\}$ such that $r \in C_i$), and to derive an arithmetic partition $\mathcal{P}^{(i_t)}$ of $\mathbb{Z}/(l/\ell)\mathbb{Z}$ for $t \in \{0, 1, \ldots, \ell - 1\}$ such that vertices in C_{i_t} on the cycle that lie in the same block of $\mathcal{P}^{(i_t)}$ have the same tree above them in Γ_f. Formally, we may express this via a labeling function $\mathrm{lab}_{i_t} : \mathcal{P}^{(i_t)} \to \{0, 1, \ldots, N\}$ that maps each block of $\mathcal{P}^{(i_t)}$ to its associated rooted tree number.

Now, the description of the connected component of Γ_f containing r obtained this way is *not* an injective encoding of its isomorphism type. That is, isomorphic connected components may end up getting different descriptions, and it appears to be a nontrivial computational problem to decide efficiently whether two given descriptions pertain to the same isomorphism type. The purpose of this section is to discuss this open problem in more detail, reducing it to some concrete questions to be answered.

Of course, in the actual implementation of our algorithm for Problem 3, each of the arithmetic partitions $\mathcal{P}^{(i_t)}$ mentioned above is expressed through a spanning congruence sequence, of length $\bar{m}_{i_t}$ say, and lab_{i_t} may be expressed through a function $\{\emptyset, \neg\}^{\bar{m}_{i_t}} \to \{-1, 0, 1, \ldots, N\}$, where -1 is a dummy value to be assigned to a logical sign tuple $\vec{\nu}$ if the associated set $\mathcal{B}(\mathcal{P}^{(i_t)}, \vec{\nu})$ is empty. By abuse of notation, we also call this function lab_{i_t}. We give a special name to ordered pairs such as $(\mathcal{P}^{(i_t)}, \mathrm{lab}_{i_t})$.

Definition 6.2.1. Let $m, N \in \mathbb{N}_0$ with $m > 0$. An *N-labeled arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$* is a pair $(\mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K), \mathrm{lab})$ consisting of an arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ with a fixed spanning congruence sequence and a so-called labeling function $\mathrm{lab} : \{\emptyset, \neg\}^K \to \{-1, 0, 1, \ldots, N\}$ such that $\mathrm{lab}(\vec{\nu}) = -1$ if and only if $\mathcal{B}(\mathcal{P}, \vec{\nu}) = \emptyset$.

In order to characterize when two compact descriptions obtained by the algorithm for Problem 3 represent isomorphic connected components, let us first consider the special case where $\ell = 1$. Then we only need to worry about the coset C_i and the associated N-labeled arithmetic partition $(\mathcal{P}^{(i)}, \mathrm{lab}_i)$ of $\mathbb{Z}/l\mathbb{Z}$.

We need to understand how applying a cyclic shift to the associated sequence of rooted tree isomorphism types affects the N-labeled arithmetic partition. Let $\mathfrak{t}$ be the translation $x \mapsto x + 1$ of $\mathbb{Z}/l\mathbb{Z}$. It generates a cyclic subgroup $\mathfrak{T}$ of order l of $\mathrm{Sym}(\mathbb{Z}/l\mathbb{Z})$, namely the image of the regular representation of $\mathbb{Z}/l\mathbb{Z}$ on itself. This group $\mathfrak{T}$ also acts naturally on the power set of $\mathbb{Z}/l\mathbb{Z}$ via

$$M^t := t(M) = \{y^t : y \in M\}.$$

In the same manner, this leads to an action of $\mathfrak{T}$ on the power set of the power set of $\mathbb{Z}/l\mathbb{Z}$ (i.e., an action which transforms families of subsets of $\mathbb{Z}/l\mathbb{Z}$ into other such families), and this action restricts to one on the set of all arithmetic partitions of $\mathbb{Z}/l\mathbb{Z}$. Indeed, if $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$ is an arithmetic partition of $\mathbb{Z}/l\mathbb{Z}$, then $\mathcal{P}^{\mathfrak{t}}$, the partition of $\mathbb{Z}/l\mathbb{Z}$ obtained by shifting all blocks of $\mathcal{P}$ to the right by one unit, is just the arithmetic partition $\mathfrak{P}(x \equiv \mathfrak{b}_j + 1 \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$ of $\mathbb{Z}/l\mathbb{Z}$.

If we assume that $\mathcal{P}$ is N-labeled and that each block of $\mathcal{P}^t$, where $t \in \mathfrak{T}$, carries the same label as the block of $\mathcal{P}$ it is shifted from, we finally get an action of $\mathfrak{T}$ on the set of N-labeled arithmetic partitions of $\mathbb{Z}/l\mathbb{Z}$, which is useful for our characterization. Formally, this action is defined via

$$\begin{aligned}&(\mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K), \mathrm{lab})^{\mathfrak{t}^n}\\ &\quad := (\mathfrak{P}(x \equiv \mathfrak{b}_j + n \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K), \mathrm{lab}).\end{aligned}$$

In the special case "$\ell = 1$" we are currently discussing, applying a right cyclic shift of n units to the cyclic sequence of rooted tree isomorphism types associated with the N-labeled arithmetic partition $(\mathcal{P}^{(i)}, \mathrm{lab}_i)$ corresponds to replacing $(\mathcal{P}^{(i)}, \mathrm{lab}_i)$ by $(\mathcal{P}^{(i)}, \mathrm{lab}_i)^{\mathfrak{t}^n}$. This motivates the following definition.

Definition 6.2.2. Let $m, N \in \mathbb{N}_0$ with $m > 0$, and let $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ be N-labeled arithmetic partitions of $\mathbb{Z}/m\mathbb{Z}$. These partitions are *equivalent* if there is a $t \in \mathfrak{T}$ such that $(\mathcal{P}', \mathrm{lab}')^t = (\mathcal{P}, \mathrm{lab})$. In that case, the smallest positive integer n such that $(\mathcal{P}', \mathrm{lab}')^{\mathfrak{t}^n} = (\mathcal{P}, \mathrm{lab})$ is the *translation number of* $(\mathcal{P}, \mathrm{lab})$ *and* $(\mathcal{P}', \mathrm{lab}')$. If $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ are not equivalent, then we define their translation number to be ∞.

We observe that translation numbers are not symmetric in their two arguments. Rather, if $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ are equivalent, then the translation number of $(\mathcal{P}', \mathrm{lab}')$ and $(\mathcal{P}, \mathrm{lab})$ is the difference of the stabilizer order

$$|\mathrm{Stab}_{\mathfrak{T}}((\mathcal{P}, \mathrm{lab}))| = |\{t \in \mathfrak{T} : (\mathcal{P}, \mathrm{lab})^t = (\mathcal{P}, \mathrm{lab})\}| = |\mathrm{Stab}_{\mathfrak{T}}((\mathcal{P}', \mathrm{lab}'))|$$

and the translation number of $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$. For example, if $(\mathcal{P}, \mathrm{lab})^{\mathfrak{t}^3} = (\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')^{\mathfrak{t}} = (\mathcal{P}, \mathrm{lab})$, then $(\mathcal{P}, \mathrm{lab})^{\mathfrak{t}^2} = (\mathcal{P}', \mathrm{lab}')$.

As we mentioned just before Definition 6.2.2, in case $\ell = 1$, two N-labeled arithmetic partitions of the form $(\mathcal{P}^{(i)}, \mathrm{lab}_i)$ and $(\mathcal{P}^{(j)}, \mathrm{lab}_j)$ correspond to isomorphic connected components of Γ_f if and only if they are equivalent. We thus pose the following algorithmic problem.

Problem 6.2.3. *Find an efficient algorithm which, for given $m, N \in \mathbb{N}_0$ with $m > 0$ and N-labeled arithmetic partitions $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ of $\mathbb{Z}/m\mathbb{Z}$, computes the translation number of $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$.*

We note that by our convention on translation numbers of inequivalent N-labeled arithmetic partitions, such an algorithm could in particular be used to efficiently decide whether $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ are equivalent in the first place. Moreover, it could be used to determine the stabilizer in $\mathfrak{T}$ of a given N-labeled arithmetic partition $(\mathcal{P}, \mathrm{lab})$, because that stabilizer is generated by $\mathfrak{t}^n$, where n is the translation number of $(\mathcal{P}, \mathrm{lab})$ with itself.

We now return from "$\ell = 1$" to the general case. By the details of our identification of the l positions on the f-cycle of r with the elements in ℓ disjoint copies of $\mathbb{Z}/(l/\ell)\mathbb{Z}$ (see Section 5.2.3), it is not hard to see that in general, applying a right cyclic shift by n units to the sequence of rooted tree isomorphism types associated with the sequence $((\mathcal{P}^{(i_t)}, \mathrm{lab}_{i_t}))_{t=0,1,\ldots,\ell-1}$ corresponds to replacing that sequence with

$$\begin{aligned}&(((\mathcal{P}^{(i_j)}, \mathrm{lab}_{i_j}))_{j=0,1,\ldots,\ell-1})^{\mathfrak{t}^n}\\ &\quad := ((\mathcal{P}^{(i_j)}, \mathrm{lab}_{i_j})^{\mathfrak{t}^{n''+1}})_{j=\ell-n',\ell-n'+1,\ldots,\ell-1} \diamond ((\mathcal{P}^{(i_j)}, \mathrm{lab}_{i_j})^{\mathfrak{t}^{n''}})_{j=0,1,\ldots,\ell-n'-1},\end{aligned}$$

where $n' := n \bmod \ell$ and $n'' := (n - n')/\ell$. This defines an action of $\mathfrak{T}$ on the set of all length ℓ sequences of N-labeled arithmetic partitions of $\mathbb{Z}/(l/\ell)\mathbb{Z}$, and as for $\ell = 1$, we call two such sequences *equivalent* if they can be mapped to each other under this action. In order to decide in general whether two descriptions produced by our algorithm for Problem 3 correspond to isomorphic connected components, we need to decide whether these descriptions are equivalent in this more general sense. However, it turns out that this can be done efficiently if we have an algorithm as in Problem 6.2.3. Let us explain why.

We assume that $(\mathcal{X}_k, \mathrm{lab}^{(k)})_{k=0,1,\ldots,\ell-1}$ and $(\mathcal{Y}_k, \mathrm{Lab}^{(k)})_{k=0,1,\ldots,\ell-1}$ are length ℓ sequences of N-labeled arithmetic partitions of $\mathbb{Z}/m\mathbb{Z}$ for some $m \in \mathbb{N}^+$. They are equivalent if and only if there are $n' \in \{0, 1, \ldots, \ell - 1\}$ and $n'' \in \mathbb{Z}$ such that $((\mathcal{Y}_k, \mathrm{Lab}^{(k)}))_{k=0,1,\ldots,\ell-1}$ is equal to

$$((\mathcal{X}_t, \mathrm{lab}^{(t)})^{\mathfrak{t}^{n''+1}})_{t=\ell-n',\ell-n'+1,\ldots,\ell-1} \diamond ((\mathcal{X}_t, \mathrm{lab}^{(t)})^{\mathfrak{t}^{n''}})_{t=0,1,\ldots,\ell-n'-1}. \tag{6.1}$$

To check whether this is the case, we assume that n' is fixed (in the worst case, we need to try out $\ell \in O(d)$ values for n'). For $t = 0, 1, \ldots, \ell - 1$, we compute the number

$$\mathfrak{b}_t := \begin{cases} \text{translation number of } (\mathcal{Y}_t, \mathrm{Lab}^{(t)}) \text{ and } (\mathcal{X}_{\ell-n'+t}, \mathrm{lab}^{(\ell-n'+t)}), & \text{if } t < n', \\ \text{translation number of } (\mathcal{Y}_t, \mathrm{Lab}^{(t)}) \text{ and } (\mathcal{X}_{t-n'}, \mathrm{lab}^{(t-n')}), & \text{otherwise} \end{cases}$$

using the algorithm from Problem 6.2.3. If any of these numbers is ∞, then the chosen value of n' does not work. Otherwise, we compute $\mathfrak{a}_t$, the group order of the stabilizer of $(\mathcal{Y}_t, \mathrm{Lab}^{(t)})$ in $\mathfrak{T}$ (which here is a cyclic group of order m), using the said algorithm. The question is whether there exists $n'' \in \mathbb{Z}$ such that

$$\begin{aligned} n'' + 1 &\equiv \mathfrak{b}_0 \pmod{\mathfrak{a}_0}, \\ n'' + 1 &\equiv \mathfrak{b}_1 \pmod{\mathfrak{a}_1}, \\ &\vdots \\ n'' + 1 &\equiv \mathfrak{b}_{n'-1} \pmod{\mathfrak{a}_{n'-1}}, \\ n'' &\equiv \mathfrak{b}_{n'} \pmod{\mathfrak{a}_{n'}}, \\ n'' &\equiv \mathfrak{b}_{n'+1} \pmod{\mathfrak{a}_{n'+1}}, \\ &\vdots \\ n'' &\equiv \mathfrak{b}_{\ell-1} \pmod{\mathfrak{a}_{\ell-1}} \end{aligned}$$

because these congruences characterize when $((\mathcal{Y}_t, \mathrm{Lab}^{(t)}))_{t=0,1,\ldots,\ell-1}$ is equal to (6.1). Viewing this as a system of m-congruences in the single variable n'', the existence of n'' can easily be decided using Proposition 2.2.1.

We conclude this section by noting that the algorithm from Problem 6.2.3 can be used to decide whether two (N-labeled) arithmetic partitions are equal, i.e., have the same (labeled) blocks. This is because two N-labeled arithmetic partitions $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ of $\mathbb{Z}/m\mathbb{Z}$ are equal if and only if the translation number of $(\mathcal{P}, \mathrm{lab})$ and $(\mathcal{P}', \mathrm{lab}')$ equals the translation number of $(\mathcal{P}, \mathrm{lab})$ with itself. Moreover, $\mathcal{P}$ and $\mathcal{P}'$ are equal if and only if $(\mathcal{P}, \mathbf{0})$ and $(\mathcal{P}', \mathbf{0})$ are equal, where (in each of the two cases) $\mathbf{0}$ denotes the constantly zero labeling function. Still, the algorithmic problem of verifying whether two given arithmetic partitions of $\mathbb{Z}/m\mathbb{Z}$ are equal is interesting in its own right, and it may admit an efficient algorithmic solution even if Problem 6.2.3 does not, so we pose it separately.

Problem 6.2.4. *Find an efficient algorithm which, for given $m \in \mathbb{N}^+$ and (spanning m-congruence sequences of) arithmetic partitions $\mathcal{P}$ and $\mathcal{Q}$ of $\mathbb{Z}/m\mathbb{Z}$, decides whether $\mathcal{P} = \mathcal{Q}$.*

6.3 More problems concerning asymptotic growth rates

Let $m = p_1^{v_1} \cdots p_K^{v_K}$ be a positive integer with its factorization displayed. We recall from Definition 1.5 (3) that the minimal number of spanning m-congruences for an arithmetic partition $\mathcal{P}$ of $\mathbb{Z}/m\mathbb{Z}$ is called the (arithmetic) complexity of $\mathcal{P}$ and denoted by $\mathrm{AC}(\mathcal{P})$. In Remark 3.3.6, we observed that the trivial partition $\mathcal{T}_m$ of $\mathbb{Z}/m\mathbb{Z}$, all of whose blocks are singletons, satisfies $\mathrm{AC}(\mathcal{T}_m) \leq \sum_{j=1}^{K} p_j^{v_j} - K$. While this bound is equal to $m - 1$ when m is a prime power, we also observed in Remark 3.3.6 that the bound is asymptotically equivalent to $\log^2 m/(2 \log\log m)$ when m is a primorial, which leads to the question whether the actual complexity of $\mathcal{T}_m$ can be significantly smaller than that (on a suitable infinite class of values for m).

Question 6.3.1. *Is it true that*

$$\liminf_{m\to\infty} \frac{\mathrm{AC}(\mathcal{T}_m)}{\log^2 m/\log\log m} > 0?$$

We observe that $\mathcal{T}_m$ is the unique (arithmetic) partition of $\mathbb{Z}/m\mathbb{Z}$ that achieves the maximum possible number of blocks, m. One may ask more generally for nontrivial bounds that relate the arithmetic complexity of an arithmetic partition $\mathcal{P}$ of $\mathbb{Z}/m\mathbb{Z}$ with its number of blocks. Trivially, the number of distinct blocks of $\mathcal{P}$ is at most $2^{\mathrm{AC}(\mathcal{P})}$, and this bound is attained if $\mathrm{AC}(\mathcal{P}) \in \{0, 1\}$. In fact, for any given value $k \in \mathbb{N}_0$, there is an $m \in \mathbb{N}^+$ and an arithmetic partition $\mathcal{P}$ of $\mathbb{Z}/m\mathbb{Z}$ such that $\mathrm{AC}(\mathcal{P}) = k$ and $\mathcal{P}$ has 2^k distinct blocks: simply let m be the k-th primorial $p_k\# = p_1 p_2 \cdots p_k$, where p_j denotes the j-th smallest prime number, and

$$\mathcal{P} := \mathfrak{P}(x \equiv 0 \pmod{p_j} : j = 1, 2, \ldots, k).$$

However, once $\mathrm{AC}(\mathcal{P})$ becomes sufficiently large with respect to m, the number of blocks of $\mathcal{P}$ falls behind $2^{\mathrm{AC}(\mathcal{P})}$; at latest, this happens once $\mathrm{AC}(\mathcal{P}) > \log_2 m$, because the block count of $\mathcal{P}$ cannot be larger than m. In the example we just gave, where $m = p_k\#$, we have $\mathrm{AC}(\mathcal{P}) = k \sim (\log m/\log\log m)$, which motivates the following open problem.

Problem 6.3.2. *Either find functions $f, g : [0, \infty) \to [0, \infty)$ such that*

(1) $(\log x/\log\log x) \lesssim f(x) \in o(\log x)$,

(2) $g(x) \in o(2^x)$, *and*

(3) *for every positive integer m and every arithmetic partition $\mathcal{P}$ of $\mathbb{Z}/m\mathbb{Z}$ with $\mathrm{AC}(\mathcal{P}) \geq f(m)$, the number of blocks of $\mathcal{P}$ is at most $g(\mathrm{AC}(\mathcal{P}))$,*

or prove that such functions do not exist.

In Section 5.3.2, we took note of a potential obstacle to using tree necklace lists to give a general, efficient algorithm for deciding whether the functional graphs of

two given generalized cyclotomic mappings of $\mathbb{F}_q$ are isomorphic. Namely, it could be that even when their index is fixed, generalized cyclotomic mappings f have too many distinct isomorphism types of connected components in their functional graphs. Specifically, we pose the following problem.

Problem 6.3.3. *Prove or disprove that for every $d \in \mathbb{N}^+$, there is a constant $c = c(d)$ such that for every prime power q and every index d generalized cyclotomic mapping f of $\mathbb{F}_q$, the number of distinct isomorphism types of connected components of Γ_f is in $O(\log^c q)$.*

We observe that for $d = 1$, all rooted trees above *non-zero* f-periodic points are isomorphic; see Theorem 2.1.5, noting that $\Gamma_{f_{|\mathbb{F}_q^*}} \cong \Gamma_{A_0}$, unless f is constantly zero, in which case the statement in question is vacuously true. Therefore, for $d = 1$, Problem 6.3.3 is equivalent to proving or disproving that the number of distinct cycle lengths of an affine map of $\mathbb{Z}/(q-1)\mathbb{Z}$, where q ranges over all prime powers, is bounded from above by some fixed polynomial in $\log q$. Even this appears to be an open problem, in spite of Remark 5.3.2.4.

In view of Proposition 5.3.1.4, we accept that an arbitrarily small but positive asymptotic fraction of prime powers q needs to be excluded in order for the algorithms from Section 5.3.2 to be efficient. In this context, we note the following problem related to Problem 6.3.3.

Problem 6.3.4. *For $d \in \mathbb{N}^+$ and $c > 0$, we denote by $\varepsilon(d, c)$ the asymptotic proportion of all prime powers q with $d \mid q-1$ for which there exists an index d generalized cyclotomic mapping f of $\mathbb{F}_q$ such that Γ_f has more than $\log^c q$ distinct isomorphism types of connected components. Prove or disprove that as d is fixed and $c \to \infty$, one has $\varepsilon(d, c) \to 0$.*

Of course, a proof of the assertion in Problem 6.3.3 also yields a proof of the assertion in Problem 6.3.4. On the other hand, if the assertion in Problem 6.3.3 is false (which the authors believe is the case), the one in Problem 6.3.4 may still be true, thus allowing one to solve the isomorphism problem for functional graphs of generalized cyclotomic mappings of a fixed index efficiently in "most" cases (only having to exclude an arbitrarily small positive asymptotic fraction of cases if one accepts a suitably large degree in the poly-log bound).

6.4 Extension to other coset-wise affine functions

An index d generalized cyclotomic mapping f of $\mathbb{F}_q$, given in cyclotomic form (1.1), such that all a_i and r_i are non-zero restricts to a function $\mathbb{F}_q^* \to \mathbb{F}_q^*$, which is "coset-wise affine" in the sense that its restriction to any given coset C_i of the index d subgroup C of $\mathbb{F}_q^*$ maps to another coset $C_{\bar{f}(i)}$ via an affine map of the cyclic group C

(here, we are using the general, group-theoretic sense of the word "affine map", as in Definition 2.1.15). That we can split f up into such smaller, easy to handle parts is crucial for the approach of understanding Γ_f presented in this memoir.

In this section, we aim to generalize this idea. More specifically, we replace $\mathbb{F}_q^*$ by some group G (usually, but not necessarily finite), and C by a subgroup H of G. We consider the following two notions of coset-wise affine functions.

Definition 6.4.1. Let G be a group, and let H be a subgroup of G.

(1) An *affine function* $H \to G$ is a function $H \to G$ of the form $h \mapsto h^\varphi b$ for some group homomorphism $\varphi : H \to G$ and some $b \in G$.

(2) A function $f : G \to G$ is called *H-coset-wise affine in the wide sense* if for every right coset $C = Hr_C$ of H in G, there is an affine function $A_C : H \to G$ such that $f(hr_C) = A_C(h)$ for all $h \in H$.

(3) A function $f : G \to G$ is called *H-coset-wise affine in the narrow sense* if for every right coset $C = Hr_C$ of H in G, there is an affine map A_C of H and a $t_C \in G$ such that $f(hr_C) = A_C(h)t_C$ for all $h \in H$.

In contrast to H-coset-wise affine functions in the narrow sense, an H-coset-wise affine function in the wide sense does not need to map each right coset of H to a single such coset. This makes it hard to study the behavior of H-coset-wise affine functions in the wide sense under iteration. The most celebrated example of this is the Collatz function g, corresponding to $G = \mathbb{Z}$ and $H = 2\mathbb{Z}$ and given by the coset-wise affine formula

$$g(x) = \begin{cases} x/2, & \text{if } x \in 2\mathbb{Z}, \\ 3x + 1, & \text{if } x \in 2\mathbb{Z} + 1. \end{cases}$$

On the other hand, any function $g : \mathbb{Z} \to \mathbb{Z}$ such that g agrees, on each coset $k + n\mathbb{Z}$ of the index n subgroup $n\mathbb{Z}$, with an affine function

$$x \mapsto a^{(k)}x + b^{(k)}$$

with *integer* coefficients $a^{(k)}$ and $b^{(k)}$, is $n\mathbb{Z}$-coset-wise affine in the narrow sense and thus amenable to the ideas mentioned in the first paragraph of this section. Henceforth, we restrict our attention to H-coset-wise affine functions in the narrow sense, which we simply call *H-coset-wise affine functions* for short.

An important special case is when $G = (\mathbb{F}_q, +)$ and H is an $\mathbb{F}_p$-subspace of G, for which this class of functions was already considered in [14]. We expect our approach for understanding functional graphs of generalized cyclotomic mappings to work mostly analogously for H-coset-wise affine functions, with one big caveat: in the proof of Lemma 2.2.2 (our "master lemma"), we made essential use of the equivalence of statements (1) and (3) in Proposition 2.2.1. In the more general group-theoretic context of the current section, this equivalence needs to be replaced by the following property of the group H.

Definition 6.4.2. Let H be a group. We say that H is *pairwise congruence-consistent* if any given system of congruences over H,

$$\begin{aligned} x &\equiv h_1 \pmod{N_1}, \\ x &\equiv h_2 \pmod{N_2}, \\ &\vdots \\ x &\equiv h_K \pmod{N_K}, \end{aligned}$$

where $h_1, h_2, \ldots, h_K \in H$ and $N_1, N_2, \ldots, N_K$ are normal subgroups of H, is consistent if and only if each pair of congruences in the system is consistent.

The equivalence of statements (1) and (3) in Proposition 2.2.1 can be reformulated as "Finite cyclic groups are pairwise congruence-consistent." In the special case "$G = (\mathbb{F}_q, +)$" mentioned above, the group H is of the form $\mathbb{F}_p^n$, i.e., it is a finite elementary abelian p-group. If our approach is to work completely analogously for that case, we would need that finite elementary abelian groups are pairwise congruence-consistent. However, that is not the case, as the following result shows (noting that all abelian groups are nilpotent).

Theorem 6.4.3. *Let G be a finite nilpotent group. The following are equivalent:*

(1) *G is pairwise congruence-consistent.*

(2) *G is cyclic.*

We prove Theorem 6.4.3 at the end of this section. Before doing so, we make two more comments.

Firstly, we observe that non-cyclic finite pairwise congruence-consistent groups exist; there are both non-solvable examples, such as any non-abelian finite simple group (for trivial reasons), and solvable examples, such as $\mathrm{AGL}_1(q) = \mathbb{F}_q \rtimes \mathbb{F}_q^*$ for any prime power q. To see that the latter kind of groups are pairwise congruence-consistent, we note that $\mathrm{AGL}_1(q)$ has $\mathbb{F}_q$ as its unique minimal, nontrivial normal subgroup, so any congruence over $\mathrm{AGL}_1(q)$ has an associated congruence over the cyclic group $\mathbb{F}_q^*$ (i.e., a congruence in the classical, number-theoretic sense) such that the solution set of the congruence over $\mathrm{AGL}_1(q)$ is the full pre-image, under the canonical projection $\mathrm{AGL}_1(q) \to \mathbb{F}_q^*$, of the solution set of the associated number-theoretic congruence. In particular, if any pair of congruences in a given system of congruences over $\mathrm{AGL}_1(q)$ is consistent, the same holds true for the associated system over $\mathbb{F}_q^*$, whence that system is consistent by Proposition 2.2.1, and so the original system over $\mathrm{AGL}_1(q)$ must also have a solution.

Secondly, we pose the following two open problems, which are motivated by Theorem 6.4.3 and the discussion leading to it.

Problem 6.4.4. *Classify the finite groups that are pairwise congruence-consistent.*

Problem 6.4.5. *For important classes of finite groups that are* not *contained in the class of finite pairwise congruence-consistent groups (such as the class of finite (elementary) abelian groups), devise efficient algorithms that decide whether a given system of congruences over a group in that class is consistent.*

In order to prove Theorem 6.4.3, we first consider the following property of groups.

Definition 6.4.6. Let G be a group. We say that G has the *pairwise coset-intersection property* (or *PCIP* for short) if the following holds: for any positive integer m and any sequence $(C_1, C_2, \ldots, C_m)$ of (left or right) cosets of subgroups of G, if $C_j \cap C_k \neq \emptyset$ for all $1 \leq j < k \leq m$, then $\bigcap_{j=1}^{m} C_j \neq \emptyset$. A group satisfying the PCIP is also called a *PCIP-group* for short.

The following proposition is immediate from observing that the solution set of the congruence $x \equiv g \pmod{N}$ over the group G is the coset $gN = Ng$ of N.

Proposition 6.4.7. *Let G be a group. The following are equivalent.*

(1) *G is pairwise congruence-consistent.*

(2) *For any positive integer m and any sequence $(C_1, C_2, \ldots, C_m)$ of cosets of normal subgroups of G, if $C_j \cap C_k \neq \emptyset$ for all $1 \leq j < k \leq m$, then $\bigcap_{j=1}^{m} C_j \neq \emptyset$.*

Proposition 6.4.7 has two important consequences.

Corollary 6.4.8. *The following hold.*

(1) *Every PCIP-group is pairwise congruence-consistent.*

(2) *An abelian group satisfies the PCIP if and only if it is pairwise congruence-consistent.*

In view of Corollary 6.4.8, the following result proves Theorem 6.4.3 for *abelian* groups.

Theorem 6.4.9. *Let G be a finite group. The following are equivalent.*

(1) *G satisfies the PCIP.*

(2) *G is cyclic.*

Proof. The implication "(2)⇒(1)" holds by Proposition 2.2.1, so we focus on "(1)⇒(2)".

It is not hard to show that all subgroups and quotients of a PCIP-group are PCIP-groups themselves. A minimal counterexample to the implication "(1)⇒(2)" would thus be a finite, non-cyclic group G all of whose proper subgroups are cyclic. These groups were classified by Miller and Moreno [52] to be one of the following.

(1) $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z}$, where p is a prime;

(2) the quaternion group Q_8; or

(3) the metacyclic group

$$\mathbb{Z}/q^n\mathbb{Z} \ltimes \mathbb{Z}/p\mathbb{Z} = \langle x, y : x^p = y^{q^n} = 1, y^{-1}xy = x^r \rangle,$$

where $r \equiv 1 \pmod{q}$ and $r^q \equiv 1 \pmod{p}$, but $r \not\equiv 1 \pmod{p}$, since otherwise, the group is cyclic or isomorphic to $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z}$.

It suffices to show that none of these groups G satisfies the PCIP, which we do now, by specifying three subsets $C_j \subseteq G$ for $j = 1, 2, 3$, each of which is a left and right coset of some subgroup of G, such that the C_j intersect pairwise while $\bigcap_{j=1}^{3} C_j = \emptyset$.

- For groups of the first type, where the elements are pairs (x, y) with $x, y \in \mathbb{F}_p$, let

$$C_1 := \{(x, y) \in \mathbb{F}_p^2 : x = 1\}, \quad C_2 := \{(x, y) \in \mathbb{F}_p^2 : y = 1\},$$

and

$$C_3 := \{(x, y) \in \mathbb{F}_p^2 : x + y = 1\}.$$

- For $Q_8 = \{\pm 1, \pm i, \pm j, \pm k\}$, let

$$C_1 := \langle i \rangle j = j \langle i \rangle = \{\pm j, \pm k\}, \quad C_2 := \langle j \rangle i = i \langle j \rangle = \{\pm i, \pm k\},$$

and

$$C_3 := \langle k \rangle j = j \langle k \rangle = \{\pm i, \pm j\}.$$

- We note that the central quotient of a group of the third type is of the same form but with $n = 1$, which we may thus assume without loss of generality. Let

$$C_1 = \langle x \rangle y = y \langle x \rangle = \{y, yx, yx^2, \ldots, yx^{p-1}\},$$
$$C_2 = \langle y \rangle = \{1, y, y^2, \ldots, y^{q-1}\},$$

and

$$C_3 = \langle yx \rangle = \{1, yx, y^2x^{1+r}, y^3x^{1+r+r^2}, \ldots, y^{q-1}x^{1+r+r^2+\cdots+r^{q-2}}\}.$$

Since the y-exponent is 1 in all elements of C_1, we have $C_1 \cap C_2 = \{y\}$ and $C_1 \cap C_3 = \{yx\}$. Moreover, $C_2 \cap C_3 = \{1\}$, since C_2 and C_3 are distinct subgroups of prime order. It follows that $C_1 \cap C_2 \cap C_3 = \emptyset$, as required. ∎

Proof of Theorem 6.4.3. As in the proof of Theorem 6.4.9, the implication "(2)⇒(1)" is clear by Proposition 2.2.1, so we focus on "(1)⇒(2)".

Let G be a finite nilpotent group that is pairwise congruence-consistent. If G is abelian, then G must be cyclic by Corollary 6.4.8 (2) and Theorem 6.4.9, so we

assume (aiming for a contradiction) that G is non-abelian. Then for some prime p, the (unique) Sylow p-group S_p of G is non-abelian. Because G is the direct product of its Sylow subgroups, we find that S_p is a quotient of G. Moreover, it is not hard to prove that quotients of pairwise congruence-consistent groups are themselves pairwise congruence-consistent. Therefore, S_p is pairwise congruence-consistent.

But S_p is a non-abelian finite p-group, whence Burnside's basis theorem implies that its Frattini quotient $S_p/\Phi(S_p)$ is the finite elementary abelian p-group $\mathbb{F}_p^n$, where $n > 1$ is the minimal size of a generating set of S_p. Using again that the property of being pairwise congruence-consistent is preserved under passing to quotients, it follows that $\mathbb{F}_p^n$ is pairwise congruence-consistent. Since $\mathbb{F}_p^n$ is abelian, Corollary 6.4.8 thus implies that $\mathbb{F}_p^n$ satisfies the PCIP, which contradicts Theorem 6.4.9. ∎

6.5 Generalization to transformation graphs

This memoir is concerned with functional graphs, which are natural visualizations of individual functions on a set and are useful for understanding the long-term behavior of discrete dynamical systems. As a generalization, one may consider the situation where a dynamical system does not evolve deterministically, but for some $n \in \mathbb{N}^+$, each system state x has n possibilities (possibly with repetitions) for its successor state, occurring with different probabilities and represented by the values $g_1(x), g_2(x), \ldots, g_n(x)$ of functions $g_j : X \to X$. Let us set $\mathcal{G} := \{g_1, g_2, \ldots, g_n\}$. A first step toward studying the behavior of such a system is to understand the so-called *transformation graph* $\mathrm{TRAG}(X, \mathcal{G})$, which is defined as the edge-labeled digraph with vertex set X whose arcs are of the form $x \xrightarrow{g_j} g_j(x)$ for $x \in X$ and $j = 1, 2, \ldots, n$. The terminology "transformation graph" is from Annexstein, Baumslag and Rosenberg's paper [7]. The concept is also closely related to operands, which are actions of semigroups on sets [19, Section 11.1] (and in analogy to the terminology "group action graph" from [7], one could also call transformation graphs "operand graphs"), and to deterministic finite automata [35, Section 2.2.1]. In fact, $\mathrm{TRAG}(X, \mathcal{G})$ is like a deterministic finite automaton with state set X and input symbol set $\mathcal{G}$, but without declared start and accept states.

We observe that except for the edge labels, $\mathrm{TRAG}(X, \{g\})$ is the same as the functional graph Γ_g in our notation. As noted in [7, beginning of Section 2.1], one may also consider a simple (i.e., no multiple arcs $x \to x'$ for given $x, x' \in X$), unlabeled version of $\mathrm{TRAG}(X, \mathcal{G})$, which we denote by $\mathrm{STRAG}(X, \mathcal{G})$. Of course, for a given set X, any digraph with vertex set X in which each vertex has positive outdegree (including possibly ∞) is of the form $\mathrm{STRAG}(X, \mathcal{G})$ for a suitable non-empty $\mathcal{G} \subseteq X^X$.

It would be interesting to know whether the methods developed in our memoir could be extended to deal with graphs of the form $\mathrm{TRAG}(\mathbb{F}_q, \mathcal{F})$ and $\mathrm{STRAG}(\mathbb{F}_q, \mathcal{F})$,

where $\mathcal{F}$ is a set of generalized cyclotomic mappings of $\mathbb{F}_q$, say of a common, small index d (which also covers some cases where the index is not uniform, because if f_j for $j = 1, 2, \ldots, n$ is a generalized cyclotomic mapping of $\mathbb{F}_q$ of index d_j, then each f_j also has index $\mathrm{lcm}(d_1, d_2, \ldots, d_n)$). Specifically, we pose the following problems.

Problem 6.5.1. *For a given prime power q and set $\mathcal{F}$ of index d generalized cyclotomic mappings of $\mathbb{F}_q$, devise efficient algorithms (say of runtime polynomial in $\log q$ for fixed d) that find*

(1) *a compact parametrization of the connected components of* $\mathrm{TRAG}(\mathbb{F}_q, \mathcal{F})$ *(equivalently, of* $\mathrm{STRAG}(\mathbb{F}_q, \mathcal{F})$*) by representative vertices, and*

(2) *a compact description of the isomorphism type of a connected component of* $\mathrm{TRAG}(\mathbb{F}_q, \mathcal{F})$*, respectively of* $\mathrm{STRAG}(\mathbb{F}_q, \mathcal{F})$*, given by a vertex in the image of the parametrization from point (1).*

To the authors' knowledge, this is an open problem even for $d = 1$ and $|\mathcal{F}| = 2$ (i.e., when considering transformation graphs that are each based on two monomial functions over $\mathbb{F}_q$).

Problem 6.5.2. *For some classes of sets of index d generalized cyclotomic mappings of $\mathbb{F}_q$, devise efficient algorithms to decide, for sets $\mathcal{F}_1, \mathcal{F}_2$ in such a class, whether* $\mathrm{TRAG}(\mathbb{F}_q, \mathcal{F}_1) \cong \mathrm{TRAG}(\mathbb{F}_q, \mathcal{F}_2)$*, respectively,* $\mathrm{STRAG}(\mathbb{F}_q, \mathcal{F}_1) \cong \mathrm{STRAG}(\mathbb{F}_q, \mathcal{F}_2)$.

Appendix A

Tabular overview of notation and terminology

The following two tables contain all pieces of notation and terminology that appear in this memoir. We start with a rather short list of notations based on mathematical symbols in Table A.1, which would be hard to find in the much longer Table A.2, the entries of which are listed in alphabetical order (placing Latin letters before Greek letters, lowercase letters before their capital counterparts, and letters in standard font before calligraphic letters, which are in turn placed before Fraktur letters).

notation	page	additional comments
$\{0,1\}^{<\infty}$	91	the set of all finite bit strings
$\circledast$	20	a certain $\mathbb{Q}$-bilinear product over $\mathbb{Q}[x_n : n \in \mathbb{N}^+]$, originally defined by Wei and Xu, such that $\mathrm{CT}(\psi_1 \otimes \psi_2) = \mathrm{CT}(\psi_1) \circledast \mathrm{CT}(\psi_2)$
$\bigotimes_{j\in I} g_j$	15	defined when each g_j is a function $X_j \to X_j$; it is the function on $\prod_{j\in I} X_j$ given by component-wise application of the g_j
gg'	23	composition of the functions g and g' (first g, then g'); synonymous: $g' \circ g$
$\langle g_1, \ldots, g_n \rangle$	32	if $g_1, \ldots, g_n$ are elements of a group G, this denotes the subgroup of G generated by the g_j
$G = H \ltimes N$	24	expresses that G is the (internal) semidirect product of H and N
$\lvert G_1 : G_2 \rvert$	19	the index of the subgroup G_2 in G_1
$G_1 \wr G_2$	8	imprimitive permutational wreath product
$H \leq G$	32	short for "H is a subgroup of G"
$\mathfrak{X} \sim \mathfrak{X}'$	64	shorthand for $\mathrm{SF}(\mathfrak{X}) = \mathrm{SF}(\mathfrak{X}')$
$\mathcal{P} \wedge \mathcal{Q}$	49	the infimum (coarsest common refinement) of the partitions $\mathcal{P}$ and $\mathcal{Q}$; if these are arithmetic partitions, then $\mathcal{P} \wedge \mathcal{Q}$ can be obtained as the arithmetic partition spanned by the concatenation of any choices of spanning congruence sequences for $\mathcal{P}$ and $\mathcal{Q}$
x^g	23	if x is an argument of the function g, this may denote the function value $g(x)$, especially in conjunction with the composition notation gg'
$\bigotimes_{j\in I} \Gamma_j$	15	the digraph tensor product of the Γ_j
$\vec{\nu} \diamond \vec{\nu}'$	57	the concatenation of $\vec{\nu}$ and $\vec{\nu}'$

Table A.1. Notations based on mathematical symbols.

notation/ terminology	page	additional comments
a	15	a variable denoting the linear coefficient of the affine map $A : x \mapsto ax + b$ of $\mathbb{Z}/m\mathbb{Z}$
α	11	a variable used to denote the modulus of an (in)congruence
$\alpha_{i,j}$	61	used in Section 3.4 and Section 5.2.3; the modulus of the j-th spanning congruence of $\mathcal{P}_i$
$\bar{\alpha}_j$	75	(used in the example in Section 4.2); the modulus of the j-th spanning congruence of $\mathcal{Q}_{0,0}$
A	15	a variable denoting an affine map of a group (mostly $\mathbb{Z}/m\mathbb{Z}$)
$\mathrm{AC}(\mathcal{P})$	12	arithmetic complexity of $\mathcal{P}$
admissible indexing function	40	none
admissible input	91	none
admissible output	91	none
affine function (between groups)	226	none
affine map (of a group)	23	none
a_i	3	defined for $i \in \{0, 1, \ldots, d-1\}$; an element of $\mathbb{F}_q$ associated with f; $f(x) = a_i x^{r_i}$ for all $x \in C_i$
A_i	8	defined for each $i \in \{0, 1, \ldots, d-1\}$ such that $a_i \neq 0$; it is the unique affine map of $\mathbb{Z}/s\mathbb{Z}$ corresponding to $f_{\restriction C_i} : C_i \to C_{\bar{f}(i)}$ under identifying $C_i, C_{\bar{f}(i)}$ with $\mathbb{Z}/s\mathbb{Z}$ via $\iota_i, \iota_{\bar{f}(i)}$, and it is given by the formula $A_i(x) = r_i x + \frac{e_i + r_i i - \bar{f}(i)}{d}$
$\mathcal{A}_i$	45	the affine map $A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$ of $\mathbb{Z}/s\mathbb{Z}$
$\mathcal{A}_{i,h}$	54	used in Section 3.3; defined for $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$ and $h \in \mathbb{N}_0$; it is defined as $A_{i-h} A_{i-h+1} \cdots A_{i-1}$, an affine map of $\mathbb{Z}/s\mathbb{Z}$
algorithmic problem	91	none
$\bar{\mathcal{A}}_{i,p}$	118	used in Sections 5.2.1 and 5.3.3; it is the reduction $\mathcal{A}_i \bmod p^{v_p(s)}$, an affine map of $\mathbb{Z}/p^{v_p(s)}\mathbb{Z}$
$\mathcal{A}'_i$	122	the affine permutation $\mathcal{A}_i \bmod s'_i$ of $\mathbb{Z}/s'_i\mathbb{Z}$
$\mathrm{aord}(x)$	34	additive order of $x \in \mathbb{Z}/m\mathbb{Z}$ (m must be clear from context)

arc	1	none
(arithmetic) complexity	12	none
arithmetic partition	11	none
$\mathrm{Aut}(G)$	33	the automorphism group of the group G
b	15	a variable denoting the constant coefficient of the affine map $A : x \mapsto ax + b$ of $\mathbb{Z}/m\mathbb{Z}$; also used more generally for the value at 1_G of an affine map of the group G
$\mathfrak{b}$	11	variable used to denote the right-hand side of a congruence or incongruence
$\mathfrak{b}_{i,j}$	61	used in Section 3.4 and Section 5.2.3; the right-hand side of the j-th spanning congruence of $\mathcal{P}_i$
$\bar{\mathfrak{b}}_j$	75	(used in the example in Section 4.2); the right-hand side of the j-th spanning congruence of $\mathcal{Q}_{0,0}$
B	27	variable denoting a block of a (usually arithmetic) partition; cf. the notation $B_{\vec{r}}$ introduced after Problem 2.3.7
$\mathcal{B}_i$	60	used in Section 3.4; the function that maps $x \in C_i$ to the unique $\mathcal{P}_i$-block containing x
$\mathcal{B}(\mathcal{P}, \vec{v})$	12	the block of the arithmetic partition $\mathcal{P}$ associated with the logical sign tuple $\vec{v}$
B_m	12	m-th Bell number
bit operation	89	none
blow-up function	83	none
bounded Las Vegas dual complexity	92	none
bounded query complexity	92	none
bracelet	10	none
bracelet graph	10	none
BU_m	83	m-blow-up function; the unique $\mathbb{Q}$-algebra endomorphism of $\mathbb{Q}[x_n : n \in \mathbb{N}^+]$ such that $\mathrm{BU}_m(x_n) = x_{mn}$ for all $n \in \mathbb{N}^+$
C	3	the index d subgroup of $\mathbb{F}_q^*$
C^C	8	the transformation semigroup of all functions $C \to C$
child	17	synonym: successor
C_i	3	a certain subset of $\mathbb{F}_q$ defined for $i \in \{0, 1, \ldots, d\}$; namely, $C_i := \omega^i C$ if $i \in \{0, 1, \ldots, d-1\}$, and $C_d := \{0_{\mathbb{F}_q}\}$

$\mathcal{C}_{\text{class}}$	91	a classical complexity (i.e., bit operation count); used for denoting the first entry of $\vec{\mathcal{C}}^{(\text{qry})}$ or $\vec{\mathcal{C}}^{(\text{LV})}$
$\mathcal{C}_{\text{conv}}$	92	a conversion complexity (i.e., count of conversions from bits to qubits and vice versa); used for denoting the third entry of $\vec{\mathcal{C}}^{(\text{LV})}$
$\mathcal{C}_{\text{fact}}$	91	a count of (integer) factorization queries; used for denoting the second entry of $\vec{\mathcal{C}}^{(\text{qry})}$
$\mathcal{C}_{\text{fdl}}$	91	a count of (finite) field discrete logarithm queries; used for denoting the fourth entry of $\vec{\mathcal{C}}^{(\text{qry})}$
$\mathcal{C}_{\text{mdl}}$	91	a count of modular discrete logarithm queries; used for denoting the third entry of $\vec{\mathcal{C}}^{(\text{qry})}$
$\mathcal{C}_{\text{mord}}$	91	a count of modular multiplicative order queries; used for denoting the fifth entry of $\vec{\mathcal{C}}^{(\text{qry})}$
$\mathcal{C}_{\text{prt}}$	91	a count of primitive root queries; used for denoting the sixth entry of $\vec{\mathcal{C}}^{(\text{qry})}$
$\mathcal{C}_{\text{quant}}$	92	a quantum complexity (i.e., count of elementary quantum gates); used for denoting the second entry of $\vec{\mathcal{C}}^{(\text{LV})}$
$\vec{\mathcal{C}}^{(\text{LV})}$	92	variable denoting a Las Vegas dual complexity, which is itself a triple of component complexities of different kinds
$\vec{\mathcal{C}}^{(\text{qry})}$	91	variable denoting a query complexity, which is itself a 6-tuple of component complexities of different kinds
$\mathfrak{C}_{i,L}$	198	the set of all $l \in \{1, 2, \ldots, L\}$ such that l is a multiple of ℓ (the $\bar{f}$-cycle length of i) and $\gcd(s, \bar{\alpha}_{i,l} - 1) \mid \bar{\beta}_{i,l}$; these are just those $l \in \{1, 2, \ldots, L\}$ for which $\eta_{i,l}(x)$ is well defined and characterizes, for $x \in C_i$, when $f^l(x) = x$
classical complexity	89	none
complexity (of a partition)	12	none
composition (of algorithmic problems)	93	none
conj(x)	23	the conjugation by $x \in G$ on the group G, i.e., the automorphism $y \mapsto x^{-1}yx$ of G
connected component (of a functional graph)	1	none
consistent (system of m-CCs)	26	none

continued fractions coefficient	99	none
convergent	99	none
conversion complexity	90	none
coset-wise affine function	226	same as coset-wise affine function in the narrow sense
coset-wise affine function in the narrow sense	226	none
coset-wise affine function in the wide sense	226	none
CRL-list	7	short for "cycle representatives and lengths list"
CT(ψ)	3	the cycle type of ψ
cycle type	2	none
cyclic sequence associated with $\vec{x}$	9	none
cyclic sequence	9	none
cyclotomic mapping	4	none
d	3	an index of f
$\mathfrak{d}$	42	a variable denoting a divisor
$\mathfrak{d}_{i,p,\vec{u}}$	124	shorthand for $\prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,\vec{u}})}$, which is the same as $\prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})}$ and always divides $l_{i,p,\vec{u}_p}$
$\mathfrak{D}$	86	a variable denoting a compact description of a finite directed rooted tree isomorphism type with respect to a fixed recursive tree description list
dihedral sequence	10	none
directed rooted tree	2	we assume that all arcs are oriented toward the root

discrete logarithm (in a group)	103	none
discrete dynamical system	1	none
distribution number	28	none
dual algorithm	88	none
dual digraph	17	none
dual model	88	none
e	20	used for exponents (degrees) of variables in cycle types; needs to be distinguished from the notation e_i
e_i	8	defined for those $i \in \{0, 1, \ldots, d-1\}$ such that $a_i \neq 0$; it is the discrete logarithm of a_i with base ω, i.e., $a_i = \omega^{e_i}$
E	17	variable denoting the edge (arc) set of a digraph
E^{-1}	17	the inverse relation of E
$E(\vec{v}, J)$	28	a certain conjunction of divisiblity conditions
equivalent labeled arithmetic partitions	221	none
equivalent sequences of labeled arithmetic partitions	222	none
equivalent systems of m-CCs	26	none
Expand(Δ)	63	the "expanded version" of the finite edge-weighted directed rooted tree Δ
f	3	a generalized cyclotomic mapping of $\mathbb{F}_q$
f_{per}	46	the restriction of f to $\bigcup_{i \in \mathrm{per}(\bar{f})} C_i$
$\bar{f}$	8	the function on $\{0, 1, \ldots, d\}$ induced by f, defined implicitly via $f(C_i) \subseteq C_{\bar{f}(i)}$
$\overline{f_{\mathrm{per}}}$	46	the induced function of f_{per}, which is equal to $\bar{f}_{\vert \mathrm{per}(\bar{f})}$
$\mathfrak{f}$	31	a variable denoting a fixed point; in Proposition 2.3.6 and discussions based on it, this is a specifically defined fixed point of an affine map of a finite cyclic group
$\mathfrak{f}_i$	146	the unique periodic point of $\mathcal{A}_i \bmod s_i''$
$\mathfrak{f}_{i,p}$	120	defined whenever $\mathcal{A}_{i,p}$ has a fixed point; a certain fixed point of $\mathcal{A}_{i,p}$, given by the formula in Proposition 2.3.6

$\mathcal{F}$	231	a set of generalized cyclotomic mappings over a common finite field
$\mathbb{F}_q$	1	the finite field of size q
finite dynamical system	1	none
functional graph	1	none
g	1	a function $X \to X$
G	23	variable denoting an abstract group
$\mathcal{G}$	230	a set of functions $X \to X$
generalized cyclotomic mapping	3	none
$\mathrm{Good}_{\vec{r}}(\mathcal{I})$	40	the set of all $\mathcal{I}$-good tuples (associated with $\vec{r}$ or, rather, with the cycle length tuple $(l_j(\vec{r}))_{j=1,2,\ldots,n}$) for the $\vec{r}$-admissible indexing function $\mathcal{I}$
good tuple	40	none
$g_{\mid Y}$	7	the restriction of g to the subset $Y \subseteq X$
h_i	51	defined for $i \in \{0, 1, \ldots, d-1\}$; if i is $\bar{f}$-transient, then $h_i = \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i))$; if i is $\bar{f}$-periodic, then $h_i = \infty$
$h'_{i,k}$	134	defined when $i < d$ is $\bar{f}$-periodic and $k \in \mathbb{Z}$; it is the smallest positive integer h' such that $\gcd(\prod_{t=0}^{h'-1} \alpha_{i_{k-h'+t}}, s) = \gcd(\prod_{t=0}^{h'-2} \alpha_{i_{k-h'+t}}, s)$; one has $H_i = \max\{h'_{i,k} : k = 0, 1, \ldots, \ell - 1\}$
$\mathfrak{h}(x)$	52	a certain technical parameter from Section 3.3; when $x \in C_i$ for an $\bar{f}$-periodic $i < d$, then our understanding of $\mathrm{Tree}_{\Gamma_f}(x)$ is gained by recursion on $\mathfrak{h}(x)$ (after dealing with $\bar{f}$-transient indices i and $i = d$ first)
H_i	52	the maximum height of the rooted trees in Γ_{per} above f-periodic vertices in one of the cosets C_{i_t} for $t \in \mathbb{Z}$
$\bar{H}$	165	the maximum tree height in $\Gamma_{\bar{f}}$
$\mathcal{H}_i$	66	defined for $i < d$ if $\bar{f}$ is a permutation; it is the common tree height above f-periodic vertices in C_i; we note that $H_i = \max_{t \in \mathbb{Z}} \mathcal{H}_{i_t}$
$\mathfrak{S}$	135	shorthand for $\max\{H_i : i \in \mathrm{per}(\bar{f}) \setminus \{d\}\}$
height	18	attribute of a finite directed rooted tree, denoting the maximum length of a directed path in it

$\mathrm{ht}(\Delta)$	18	the height of Δ
hyperkernel	23	none
i	3	an index that can range over $\{0, 1, \ldots, d\}$
i_t	45	a notation defined for $i \in \mathrm{per}(\bar{f})$ and $t \in \mathbb{Z}$; shorthand for $(\bar{f}_{\lvert \mathrm{per}(\bar{f})})^t(i)$; in particular, $(i_0, i_1, \ldots, i_{\ell-1})$ is the $\bar{f}$-cycle of $i = i_0$
i'	52	shorthand for i_{-1}
$\mathfrak{i}$	180	a variable denoting an injective function
I	15	a variable denoting an index set
I_i	61	used in Section 3.4; the set of those $j \in \{1, 2, \ldots, m_i\}$ for which the truth value of $x \equiv \mathfrak{b}_{i,j} \pmod{\mathfrak{a}_{i,j}}$ is constant along the $\mathcal{A}_i$-cycle of r_i modulo s
$\mathcal{I}$	40	variable denoting an $\vec{r}$-admissible indexing function (for some $\vec{r}$)
$\mathcal{I}_i$	124	a certain function $\mathfrak{P}_i' \to \mathfrak{P}_i$; if $\mathcal{I}_{i,\vec{\mathfrak{u}}}(\mathfrak{p})$ only has one distinct value for $\vec{\mathfrak{u}} \in \overline{Y}_i$, then $\mathcal{I}_i(\mathfrak{p})$ is that value, otherwise $\mathcal{I}_i(\mathfrak{p})$ is the unique value distinct from $\mathfrak{p}$ which $\mathcal{I}_{i,\vec{\mathfrak{u}}}(\mathfrak{p})$ assumes
$\mathcal{I}_{i,\vec{\mathfrak{u}}}$	122	a certain $\vec{r}_i(\vec{\mathfrak{u}})$-admissible indexing function, extended such that its domain of definition is $\mathfrak{P}_i'$
$\mathfrak{T}$	18	variable denoting an isomorphism type of finite directed rooted trees (either edge-weighted or not, depending on the context)
$\mathfrak{T}^+$	49	the rooted tree isomorphism type obtained by attaching a single copy of $\mathfrak{T}$ to a new root
$\mathfrak{T}_i$	82	(used in Section 4.3) shorthand for $\mathrm{Tree}_{\Gamma_{\bar{f}}}(i)$
$\mathcal{I}$-good tuple	40	none
induced by f	8	attribute referring to the function $\bar{f}$
$\mathrm{im}(g)$	49	the image (synonymously, range) of the function g
infimum (of partitions)	49	none
$\mathrm{inv}_n(u)$	29	the multiplicative inverse modulo n of $u \in \mathbb{Z}$ with $\gcd(u, n) = 1$
isomorphism of vertex-labeled digraphs	9	none

isomorphism of finite edge-weighted directed rooted trees	63	none
iterated pre-image	2	none
J	28	variable denoting an index set
$J_+(\vec{\nu})$	27	the set of indices j such that the j-th entry of $\vec{\nu}$ is $\emptyset$
$J_-(\vec{\nu})$	27	the set of indices j such that the j-th entry of $\vec{\nu}$ is $\neg$
$\ker^{(k)}(\varphi)$	19	defined for $k \in \mathbb{N}_0$ and a group endomorphism φ of G; it is the normal subgroup $\{x \in G : \varphi^k(x) = 1_G\}$ of G
$\mathcal{J}_{i,p,k}$	123	one of up to five subintervals of the range for u_p such that $\nu_p(l_{i,p,\vec{u}_p})$ is given by a simple formula in u_p whenever u_p lies in a fixed $\mathcal{J}_{i,p,k}$
$\mathcal{J}'_{i,p,k}$	123	a subinterval of $\mathcal{J}_{i,p,k}$ such that $\mathcal{I}_{i,\vec{\bar{u}}}(p) = p$ if and only if $u_p \in \bigcup_{k=1}^{\mathfrak{m}_{i,p}} \mathcal{J}'_{i,p,k}$
k_p	124	(in Section 5.2.1) the p-labeled entry of $\vec{k}$ (an element of $\mathbb{Z}/l_{i,\vec{\bar{u}}}\mathbb{Z}$)
$\vec{k}$	124	(in Section 5.2.1) variable denoting an $\mathcal{I}_{i,\vec{\bar{u}}}$-good tuple
k'_p	125	(in Section 5.2.1) the p-labeled entry of $\vec{k'}$ (an element of $\mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{\bar{u}}})\mathbb{Z}$)
$\vec{k'}$	125	a tuple, ranging over $\prod_{\mathfrak{p}\in\mathfrak{P}_i} \mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{\bar{u}}})\mathbb{Z}$, which for a fixed $\vec{\bar{u}} \in \bar{Y}_i$ serves as the parameter (argument) of a certain bijective parametrization of $\mathrm{Good}_{\vec{r}_i(\vec{\bar{u}})}(\mathcal{I}_{i,\vec{\bar{u}}})$, the set of good tuples for the $\vec{r}_i(\vec{\bar{u}})$-admissible indexing function $\mathcal{I}_{i,\vec{\bar{u}}}$
$K'_{i,\vec{\bar{u}}}$	125	the set $\prod_{\mathfrak{p}\in\mathfrak{P}_i} \mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{\bar{u}}})\mathbb{Z}$, from which $\vec{k'}$ stems
l	7	a cycle length of f or g
l_{bit}	172	a bit length
$l_{i,j}$	61	used in Section 3.4 and Section 5.2.3; the cycle length of r'_i under $\mathcal{A}_i$ modulo $\mathfrak{a}_{i,j}$
$l_{i,\vec{\bar{u}}}$	122	the cycle length $\mathrm{lcm}\{l_{i,p,\vec{u}_p} : p \in \mathfrak{P}_i\}$ of $\bigotimes_{p\in\mathfrak{P}_i} \mathcal{A}_{i,p}$ associated with $\vec{\bar{u}}$
$l_{i,p,\vec{u}_p}$	122	the "component cycle length" $\mathrm{proj}_2(\mathrm{par}'_{i,p}(\vec{u}_p))$
ℓ	45	variable denoting a cycle length of $\bar{f}$, as opposed to a cycle length of f, which is denoted by l instead; if $i \in \{0, 1, \ldots, d\}$ has been fixed, then ℓ is *always* the $\bar{f}$-cycle length of i

$\mathfrak{l}$	146	shorthand for $\log_\omega(r)$, where $r \in \mathbb{F}_q$ is an f-periodic point representing a connected component of Γ_f
$\mathfrak{l}_{i,j}$	61	used in Section 3.4 and Section 5.2.3; it denotes the affine discrete logarithm value $\log_{\mathcal{A}_i}^{(\mathfrak{a}_{i,j})}(r_i, \mathfrak{b}_{i,j})$
L	197	used in Section 5.3.3, denoting a maximal cycle length
L_i	125	the smallest non-negative integer such that $\gcd(\overline{\alpha}_i^{L_i}, s) = \prod_{p \mid \gcd(\overline{\alpha}_i, s)} p^{\kappa_p}$, used in the formula for the unique periodic point of $\mathcal{A}_i \bmod (s/s_i')$
$\mathfrak{L}$	7	variable denoting a CRL-list
$\overline{\mathfrak{L}}$	45	a CRL-list of $\bar{f}$
$\mathfrak{L}^{(1)}$	39	the set of first entries of pairs in the CRL-list $\mathfrak{L}$
$\mathfrak{L}_i$	46	a CRL-list of the restriction of f to $\bigcup_{t=0}^{\ell-1} C_{i_t}$
$\mathfrak{L}_i'$	45	a CRL-list of $\mathcal{A}_i$
$\mathfrak{L}_{i,p}'$	118	a CRL-list of $\overline{\mathcal{A}}_{i,p}$ in which all specified cycle lengths are fully factored; for the precise definition, see Table 5.2
$\mathfrak{L}_i''$	122	a CRL-list of $\mathcal{A}_i'$
$\mathfrak{L}(p^v, a)$	33	a certain CRL-list of the automorphism μ_a of $\mathbb{Z}/p^v\mathbb{Z}$, defined in Table 2.1 and (for $p > 2$) depending on a choice of primitive root $\mathfrak{r}$ modulo p^v
$\mathfrak{L}(p^v, a, b)$	37	a certain CRL-list of the affine permutation $x \mapsto ax + b$ of $\mathbb{Z}/p^v\mathbb{Z}$, defined in Table 2.2 and (for $p > 2$) depending on a choice of primitive root $\mathfrak{r}$ modulo p^v
$\mathfrak{L}$	91	variable denoting an algorithmic problem
$\mathfrak{L}_{\text{in}}$	91	the set of admissible inputs for the algorithmic problem $\mathfrak{L}$; formally, this is the domain of definition of the function $\mathfrak{L}$
$\mathfrak{L}\mathfrak{L}'$	93	the composition of the algorithmic problems $\mathfrak{L}$ and $\mathfrak{L}'$ (first $\mathfrak{L}$, then $\mathfrak{L}'$); synonymous notation: $\mathfrak{L}' \circ \mathfrak{L}$
$\mathfrak{L}' \circ \mathfrak{L}$	93	see $\mathfrak{L}\mathfrak{L}'$
lab	220	variable denoting a labeling function (i.e., the second entry of a labeled arithmetic partition)
lab_i	220	a certain labeling function for the arithmetic partition $\mathcal{P}^{(i)}$, used in two slightly different meanings (see also the paragraph before Definition 6.2.1)
Lab	222	variable denoting a labeling function (like lab)
labeled arithmetic partition	220	none
Las Vegas algorithm	88	none

Layer_h	116	the set of those $i \in \{0, 1, \ldots, d-1\}$ for which $h_i = h$
left-regular representation	23	none
$\log_a^{(m)}(x)$	41	the discrete logarithm modulo m of x with base a
$\log_A^{(m)}(x, y)$	41	an "affine discrete logarithm", formally defined at the beginning of Section 2.4
$\log_x(y)$	103	if x and y are elements of a group G, this denotes the discrete logarithm of y with base x (defined to be ∞ if y is not a power of x)
m	1	a positive integer; used as a general modulus
m_i	50	defined for $i \in \{0, 1, \ldots, d-1\}$; it denotes the length of a fixed spanning s-congruence sequence of $\mathcal{P}_i$
$\bar{m}_i$	220	length of the standard spanning congruence sequence for $\mathcal{P}^{(i)}$; in the notation of Section 3.4, one has $\bar{m}_i = m_i - \lvert I_i \rvert$
m_p	104	the so-called p-part of $m \in \mathbb{N}^+$; defined as $p^{\nu_p(m)}$
$m_{p'}$	103	the so-called p'-part of $m \in \mathbb{N}^+$; defined as $m/p^{\nu_p(m)}$
m'	31	the product of all prime powers $p^{\nu_p(m)}$, where p does *not* divide a certain other integer that is clear from context (usually the linear coefficient of a certain affine map of $\mathbb{Z}/m\mathbb{Z}$, whereas on page 101, it is the integer 2); not to be confused with the notation i'
m''	31	the quotient m/m'
$\mathfrak{m}$	173	variable denoting a count of isomorphism types of connected components of a functional graph
$\mathfrak{m}_i$	203	a function, used in the proof of Theorem 5.3.3.4, which encodes the block sizes of $\mathcal{W}_{i,L}$
$\mathfrak{m}_{i,p}$	123	the number of intervals $\mathcal{J}_{i,p,k}$ $(k = 1, 2, \ldots, \mathfrak{m}_{i,p})$
M	48	variable denoting a subset of a universal set (usually $\mathbb{Z}/m\mathbb{Z}$ or $\mathbb{F}_q$)
M^c	28	the complement set of M (in its respective universal set)
m-CC	11	short for "m-congruential condition"
m-(in)congruence	11	an (in)congruence whose modulus divides m
merging (of sorted arrays)	95	none
$\mathrm{minperl}(\vec{x})$	172	minimal period length of the finite sequence $\vec{x}$
Monte Carlo algorithm	98	none

$\mathrm{mpe}(m)$	22	the maximum prime exponent of m, i.e., $\mathrm{mpe}(m) = \max_p \nu_p(m)$
multiple of a rooted tree isomorphism type	49	none
n_i	53	the length of a fixed spanning s-congruence sequence for $\mathcal{R}_i$
$\bar{n}_i$	165	used in the proof of Lemma 5.3.2.2 (2); it denotes the unique $n \in \mathcal{N}$ such that $\mathfrak{T}_n = \mathrm{Tree}_{\Gamma_f}(x)$ for each $x \in C_i$
$\bar{n}_{i,k}$	168	used in the proof of Lemma 5.3.2.2 (4); it denotes the unique $n \in \mathcal{N}$ such that $\mathrm{Expand}(\mathfrak{T}_{i,k}) = \mathfrak{T}_n$ (i.e., $i \in S_{n,\mathrm{trans}}$ and $\mathrm{ht}(\mathfrak{T}_n) = k$)
$n'_i(\vec{v})$	148	a special notation used in Section 5.2.3; it denotes the unique $n \in \mathcal{N}$ such that $\vec{v}^+ \in S_{i,n}$
$\mathfrak{n}_p$	120	used in Section 5.2.1; it denotes the number of distinct prime divisors of $p-1$
$\mathcal{N}$	131	used in Sections 5.2.2 and 5.3.2; it denotes an initial segment of $\mathbb{N}_0$, consisting of those n for which the compact tree description $\mathfrak{D}_n$ has been defined (at the respective point in the algorithm in question)
$\mathfrak{N}$	174	variable denoting a tree necklace list
$\mathbb{N}^+$	3	the set of positive integers
$\mathbb{N}_0$	1	the set of non-negative integers
necklace	9	none
necklace graph	9	none
$\mathrm{nil}(\varphi)$	23	the hyperkernel of φ
$\mathcal{O}_{i,L}$	196	defined for $\bar{f}$-periodic $i \in \{0,1,\ldots,d-1\}$ and $L \in \mathbb{N}^+$; it is the set of all logical sign tuples $\diamond_{t=0}^{H_i+L-1} \vec{o}'_t \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-H_i-L+1}}}$ such that the associated block $\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1} \vec{o}'_t \diamond \vec{\xi}_{i,H_i})$ of $\mathcal{Q}_{i,H_i+L-1}$ consisting of f-periodic points is non-empty
$\mathrm{ord}(x)$	32	the order of the group element $x \in G$; the group G must be clear from context; for elements of $\mathbb{Z}/m\mathbb{Z}$, this always denotes the multiplicative order (hence is only well defined if x is a unit modulo m) – see also the notation aord
$\mathrm{ord}_n(x)$	104	the multiplicative order of x modulo n
p	15	variable denoting a prime, not necessarily the prime base of q; using p as a summation index implies that only prime indices satisfying the explicitly stated constraints should be used

$\mathfrak{p}$	123	variable used to denote a general element of $\mathfrak{P}'_i$ (to be distinguished from p used for elements of $\mathfrak{P}_i$)
$\mathfrak{p}_{p,k}$	120	the k-th prime divisor of $p-1$ (in a fixed factorization)
$P(T)$	85	a polynomial in the variable T
$\mathcal{P}$	11	variable denoting a partition, usually an arithmetic one
$\mathcal{P}_i$	11	a certain arithmetic partition of $C_i \cong \mathbb{Z}/s\mathbb{Z}$, defined in Section 3.3, which "controls" the trees above vertices in C_i
$\mathcal{P}^{(i)}$	61	used in Section 3.4 and Section 5.2.3; a certain arithmetic partition that encodes (part of) the cyclic sequence of rooted tree isomorphism types that represents the connected component of Γ_f containing r'_i
$\mathcal{P}'_i$	50	shorthand for $\mathfrak{P}'(\mathcal{P}_i, A_i)$
$\mathcal{P}_{i,h}$	56	a certain arithmetic partition of C_i such that for $x \in C_i$ with $\mathfrak{h}(x) = h$, the isomorphism type $\mathrm{Tree}_{\Gamma_f}(x)$ only depends on the $\mathcal{P}_{i,h}$-block in which x is contained
$\mathfrak{P}$	11	an operator, used in the notation $\mathfrak{P}(x \equiv \mathfrak{b}_k \pmod{\mathfrak{a}_k} : k = 1, 2, \ldots, K)$, which denotes the arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ (for implicitly clear $m \in \mathbb{N}^+$) spanned by those m-congruences
$\mathfrak{P}_i$	118	the set of all primes $p \mid s$ such that $p \nmid \overline{\alpha}_i$ (i.e., such that $\mathcal{A}_{i,p}$ is a permutation of $\mathbb{Z}/p^{v_p(s)}\mathbb{Z}$)
$\mathfrak{P}_{i,p,\vec{u}}$	124	the pre-image set $\mathcal{I}^{-1}_{i,\vec{u}}(\{p\})$
$\mathfrak{P}'_i$	122	the extended domain of definition $\mathfrak{P}_i \cup \pi(\prod_{p \in \mathfrak{P}_i}(p-1))$ of $\mathcal{I}_{i,\vec{u}}$
$\mathfrak{P}(\Theta(x))$	55	the arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ spanned by the non-negated versions of the m-CCs in $\Theta(x)$
$\mathfrak{P}'(\mathcal{P}, A)$	27	defined when $\mathcal{P}$ is an arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ with a fixed spanning congruence sequence and A is an affine map of $\mathbb{Z}/m\mathbb{Z}$; it is a certain arithmetic partition of $\mathbb{Z}/m\mathbb{Z}$ such that the intersection size $\lvert A^{-1}(\{x\}) \cap B\rvert$, where $B = \mathcal{B}(\mathcal{P}, \vec{v})$ is a fixed block of $\mathcal{P}$, is constant (equal to $\sigma_{\mathcal{P},A}(\vec{v}, \vec{v'})$) when x ranges over a fixed block $\mathcal{B}(\mathcal{P}', \vec{v'})$ of $\mathcal{P}' = \mathfrak{P}'(\mathcal{P}, A)$
pairwise congruence-consistent	227	none
pairwise coset-intersection property	228	none

par_i	117	a bijective parametrization of $\mathcal{L}_i$; for $i < d$, this is obtained from par'_i by stretching all cycle lengths by the factor ℓ (the $\bar{f}$-cycle length of i)
par'_i	118	a bijective parametrization of $\mathcal{L}'_i$ (for $i < d$)
par''_i	122	a bijective parametrization of $\mathcal{L}''_i$
$\mathrm{par}'_{i,p}$	118	a bijective parametrization of $\mathcal{L}'_{i,p}$
partition-tree register	87	none
PCIP	228	short for "pairwise coset-intersection property"
PCIP-group	228	none
$\mathrm{per}(g)$	1	the set of g-periodic points in X
period (length) (of an argument under a function)	1	none
periodic	1	synonymous uses: periodic under f; f-periodic
period length (of a finite sequence)	172	none
$\mathrm{perl}_g(x)$	1	the period (length) of $x \in X$ under g
phase inversion	90	none
$\mathrm{pperl}_g(x)$	1	the pre-period (length) of $x \in X$ under g
pre-period (length)	1	none
pre-periodic	1	none
$\mathrm{proc}_{i,h}$	47	an auxiliary (procreation) number, used in the proof of Theorem 3.2.1
$\mathrm{proc}_k(x)$	17	the k-th procreation number in Γ of $x \in \mathrm{V}(\Gamma)$; also written $\mathrm{proc}_k^{(\Gamma)}(x)$ for greater clarity
procreation number	17	none
$\mathrm{proj}_{i,L}$	197	the projection mapping each tuple in $\mathcal{O}_{i,L}$ to its initial segment of length $n_{i_0} + n_{i_{-1}} + \cdots + n_{i_{-H_i}}$
proj_j	122	defined for $j \in \{1, 2\}$, it is the (class-sized) function mapping an ordered pair to its j-th entry
q	1	a prime power

$\mathcal{Q}_{i,h}$	56	the arithmetic partition $\mathcal{P}_{i,h} \wedge \mathcal{U}_i$ of C_i; plays an auxiliary role in the proof of Proposition 3.3.4; moreover, $\mathcal{P}_i := \mathcal{Q}_{i,H_i}$
quantum complexity	88	none
query algorithm	88	none
query model	88	none
r	7	a periodic point under f or g that represents a cycle of that function; exception: the use as an exponent in "r-th order cyclotomic mapping" (cf. the notation r_i)
r_i	3	defined for $i \in \{0, 1, \ldots, d-1\}$; a non-negative integer between 0 and $q-2$ associated with f; one has $f(x) = a_i x^{r_i}$ for all $x \in C_i$
$r_{i,p}(\vec{u}_p)$	122	same as $\mathrm{proj}_1(\mathrm{par}'_{i,p}(\vec{u}_p))$, the p-labeled entry of $\vec{r}_i(\vec{u})$
r'_i	60	used in Section 3.4 and Section 5.2.3; a certain coset representative of C_i
$r'_{i,\vec{u}}(\vec{k'})$	125	the unique element of $\mathbb{Z}/s'_i\mathbb{Z}$ that is congruent to $\bar{\mathcal{A}}_{i,p}^{k'_p \mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p))$ modulo p^{κ_p} for each $p \in \mathfrak{P}_i$; it is given by the formula $\sum_{p\in\mathfrak{P}_i} \bar{\mathcal{A}}_{i,p}^{k'_p \mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p)) \frac{s'_i}{p^{\kappa_p}} \mathrm{inv}_{p^{\kappa_p}}(\frac{s'_i}{p^{\kappa_p}})$
$\vec{r}$-admissible indexing function	40	none
$\vec{r}_i(\vec{u})$	122	the element $(\mathrm{proj}_1(\mathrm{par}'_{i,p}(\vec{u}_p)))_{p\in\mathfrak{P}_i}$ of $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$ associated with $\vec{u}$; these elements are representatives for the orbits of the natural (component-wise) action of $\langle \mathcal{A}_{i,p} : p \in \mathfrak{P}_i \rangle$ on $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$
$r^{\langle\psi\rangle}$	39	the set of points on the ψ-cycle of r, a special case of the notation x^G for the orbit of $x \in X$ under the natural action of $G \leq \mathrm{Sym}(X)$
$\mathfrak{r}$	33	variable denoting a primitive root
$\mathfrak{r}_p$	120	a primitive root modulo $p^{v_p(m)}$, where $m \in \mathbb{N}^+$ is clear from context (in Section 5.2.1, $m = s$)
$\mathcal{R}_i$	53	defined for $i \in \{0,1,\ldots,d-1\}$; it is the arithmetic partition $\bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$, where $j_1, j_2, \ldots, j_K$ are the $\bar{f}$-transient pre-images of i under $\bar{f}$; if i is $\bar{f}$-transient itself, then $\mathcal{R}_i = \mathcal{P}_i$
$\mathfrak{R}$	164	variable denoting a type-I or -II tree register
recursive tree description list	86	none
right-regular representation	23	none

rigid procreation	17	none
rooted tree	2	for us, synonymous with "directed rooted tree"
rt(Δ)	18	the root of Δ
s	8	the group order of C, i.e., $s = (q-1)/d$
s_i'	122	shorthand for $\prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$
s_i''	146	shorthand for s/s_i'
$\mathfrak{s}$	40	variable denoting a shift (additive translation) of a finite abelian group; see in particular the notation $\mathfrak{s}_{\vec{r}}$ introduced after Problem 2.3.7
S_n	162	notation used in type-I and -II tree registers; it contains the information where the trees from the register occur in Γ_f
$S_{n,i}$	87	appears in the definition of a partition-tree register; it is either a specific logical sign tuple, or a set thereof, or a tuple of such sets; together, the $S_{n,i}$ encode the information which block of $\mathcal{P}_i$ corresponds to which rooted tree
$S_{n,i,h}$	87	for $i \neq d$ that is $\bar{f}$-periodic, this denotes one of the entries of the tuple $S_{n,i}$
$\overline{S}_{n,i}$	148	a special notation used in Section 5.2.3; the element-wise image of $S_{n,i}$ under $\vec{\nu}' \mapsto \vec{\nu}'^{-}$
$S_{n,\mathrm{per}}$	163	part of S_n in a type-II tree register; it stores the information for which $i \in \{0,1,\dots,d-1\}$ the rooted tree isomorphism type $\mathfrak{T}_n$ occurs above f-periodic vertices in C_i
$S_{n,\mathrm{trans}}$	163	part of S_n in a type-II tree register; it stores the information for which $i \in \{0,1,\dots,d-1\}$ the rooted tree isomorphism type $\mathfrak{T}_n$ occurs above f-transient vertices in C_i
$\mathcal{S}_{i,h}$	54	defined when $i \in \mathrm{per}(\bar{f}) \setminus \{d\}$ and $h \in \{0,1,\dots,H_i\}$; it is a certain arithmetic partition of C_i such that for $x \in C_i$ and $\mathfrak{h}(x) = h$, the isomorphism type $\mathrm{Tree}_{\Gamma_f}(x, C_{i'})$ only depends on the block of $\mathcal{S}_{i,h}$ in which x is contained
semidirect product	24	none
SF($\mathfrak{T}$)	64	simplified form of $\mathfrak{T}$
simplified	63	none
simplified form	64	none
spanned by	11	none
special type I	162	not to be confused with "type I"
special type II	162	not to be confused with "type II"

$\mathrm{Stab}_G(x)$	222	defined when G is a group acting on a set X and $x \in X$; it denotes the stabilizer of x in G
state space	1	none
$\mathrm{STRAG}(X, \mathcal{G})$	230	the underlying digraph of $\mathrm{TRAG}(X, \mathcal{G})$, which has no edge labels and no multiple edges $x \to x'$ for fixed $x, x' \in X$
successor	17	synonym: child
successor generations	17	none
sum of rooted tree isomorphism types	49	none
sum of simplified edge-weighted rooted tree isomorphism types	64	none
$\mathrm{Sym}(n)$	8	same as $\mathrm{Sym}(\{0, 1, \ldots, n-1\})$
$\mathrm{Sym}(X)$	8	symmetric group on X
synchronization	180	none
t	221	the translation $x \mapsto x + 1$ of the cyclic group $\mathbb{Z}/m\mathbb{Z}$ (m must be clear from context)
T	85	a formal variable, used for polynomial rings
$\mathcal{T}_m$	59	the trivial partition of $\mathbb{Z}/m\mathbb{Z}$ (all blocks are singletons)
$\mathcal{T}_{i,h}$	56	the arithmetic partition $\mathcal{S}_{i,h} \wedge \mathcal{U}_i$ of C_i; plays an auxiliary role in the proof of Proposition 3.3.4
$\mathfrak{T}$	221	the cyclic subgroup of $\mathrm{Sym}(\mathbb{Z}/m\mathbb{Z})$ generated by t
tensor product (of digraphs)	15	none
$\mathrm{TRAG}(X, \mathcal{G})$	230	the transformation graph associated with the set X and set $\mathcal{G}$ of functions $X \to X$
transient	1	synonymous uses: transient under f; f-transient
transition function	1	none
translation number	221	none
$\mathrm{Tree}_i(\mathcal{P}_i, B)$	61	defined when $B = \mathcal{B}(\mathcal{P}_i, \vec{v})$; the same as $\mathrm{Tree}_i(\mathcal{P}_i, \vec{v})$
$\mathrm{Tree}_i(\mathcal{P}_i, \vec{v})$	12	the rooted tree isomorphism type associated with the block $\mathcal{B}(\mathcal{P}_i, \vec{v})$ of $\mathcal{P}_i$

$\mathrm{Tree}_i(\mathcal{P}, M, \vec{\nu})$	49	defined when $\mathrm{Tree}_{\Gamma_f}(x, M)$ for $x \in C_i$ only depends on the $\mathcal{P}$-block in which x is contained (under the identification of C_i with $\mathbb{Z}/s\mathbb{Z}$ via ι_i); it denotes the common rooted tree isomorphism type $\mathrm{Tree}_{\Gamma_f}(x, M)$ for $x \in \mathcal{B}(\mathcal{P}, \vec{\nu})$
$\mathrm{Tree}_i(\mathcal{P}, \vec{\nu})$	49	shorthand for $\mathrm{Tree}_i(\mathcal{P}, \mathcal{B}(\mathcal{P}, \vec{\nu}))$; a natural extension of the notation $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu})$ to arbitrary arithmetic partitions of C_i
$\mathrm{Tree}_i(\mathcal{P}'_j, C_j, \vec{\nu}^{(\mathcal{P}'_j)})$	53	defined when $i \in \{0, 1, \ldots, d-1\}$ and j is an $\bar{f}$-transient pre-image of i under $\bar{f}$; it denotes the common isomorphism type $\mathrm{Tree}_{\Gamma_f}(x, C_j)$ for $x \in \mathcal{B}(\mathcal{P}'_j, \vec{\nu}^{(\mathcal{P}'_j)}) \subseteq C_i$
$\mathrm{Tree}_i(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{R}_i)})$	53	defined when $i \in \{0, 1, \ldots, d-1\}$ and the j_t are the $\bar{f}$-transient pre-images of i under $\bar{f}$; it denotes the common isomorphism type $\mathrm{Tree}_{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$ for $x \in \mathcal{B}(\mathcal{R}_i, \vec{\nu}^{(\mathcal{R}_i)}) \subseteq C_i$
$\mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$	57	defined when $i < d$ is $\bar{f}$-periodic and $h \in \{0, 1, \ldots, H_i\}$; a rooted tree isomorphism type to which all $\mathrm{Tree}_{\Gamma_f}(x)$ for $x \in \mathcal{B}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})}) \subseteq C_i$ with $\mathfrak{h}(x) = h$ are equal
$\mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$	57	defined when $i < d$ is $\bar{f}$-periodic and $h \in \{0, 1, \ldots, H_i\}$; a rooted tree isomorphism type to which all $\mathrm{Tree}_{\Gamma_f}(x, C_{i'})$ for $x \in \mathcal{B}(\mathcal{S}_{i,h}, \vec{\nu}^{(\mathcal{S}_{i,h})}) \subseteq C_i$ with $\mathfrak{h}(x) = h$ are equal
$\mathrm{Tree}_\Gamma(x)$	9	the tree above x in Γ
$\mathrm{Tree}_\Gamma(x, M)$	48	like $\mathrm{Tree}_\Gamma(x)$, but deleting all subgraphs $\mathrm{Tree}_\Gamma(y)$, where $y \to x$ and $y \notin M$
tree above x in Γ	9	none
tree necklace list (relative to a sequence of rooted tree isomorphism types)	173	none
tree necklace list (relative to a tree register)	174	none
type I	147	not to be confused with "special type I" or "type-I tree register"
type-I tree register	162	none
type II	147	not to be confused with "special type II" or "type-II tree register"
type-II tree register	163	none
type III	147	none

u	120	the first of up to two parameters making up $\vec{u}$
u_p	122	the first of up to two parameters making up $\vec{u}_p$
u'	120	the second of up to two parameters making up $\vec{u}$
u'_p	122	the second of up to two parameters making up $\vec{u}_p$
$\vec{u}$	120	general form of an element of the parameter set $Y_{i,p}$ (depends on p, which is suppressed in this notation; see also $\vec{u}_p$)
$\vec{u}_p$	122	the p-labeled component of $\vec{\vec{u}}$; it is an element of $Y_{i,p}$
$\vec{\vec{u}}$	122	general form of an element of the parameter set $\overline{Y}_i$
$\mathfrak{u}_{i,L}$	196	the function $\mathcal{O}_{i,L} \to \mathcal{O}_{i',L-1}$ mapping each tuple $\diamond_{t=0}^{H_i+L-1}\vec{o}'_t \in \mathcal{O}_{i,L}$ to the unique tuple $\diamond_{t=0}^{H_i+L-2}\vec{o}_t \in \mathcal{O}_{i',L-1}$ such that for each (f-periodic) point $x \in \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o}'_t \diamond \vec{\xi}_{i,H_i})$, the unique f-periodic pre-image $x^{(-1)} \in C_{i'}$ of x under f lies in the block $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \mathfrak{u}_{i,L}(\diamond_{t=0}^{H_i+L-1}\vec{o}_t) \diamond \vec{\xi}_{i',H_i})$
U_i	83	defined for $\bar{f}$-periodic i, say of cycle length ℓ; it is the union $\bigcup_{t=0}^{\ell-1} C_{i_t}$ of the blocks C_j for all indices j on the $\bar{f}$-cycle of i
U_χ	90	phase inversion operator associated with χ
$\mathcal{U}_i$	55	defined when $i \in \{0, 1, \ldots, d-1\}$ is $\bar{f}$-periodic; it is the arithmetic partition $\mathfrak{P}(\theta_{i,h}(x) : h = 1, 2, \ldots, H_i)$ of C_i, the blocks of which are the subsets of C_i consisting of points with a common $\mathfrak{h}$-value
v	15	variable denoting an exponent in a prime factorization
$v_{p,k}$	120	shorthand for $v_{\mathfrak{p}_{p,k}}(p-1)$
$v'_{i,p}$	120	shorthand for $v_p(\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i))$
$v'_{i,p,k}$	120	shorthand for $v_{\mathfrak{p}_{p,k}}(\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i))$
$v''_{i,2}$	120	shorthand for $v_2(\mathrm{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i))$
$\mathfrak{v}_i$	49	the number of vertices strictly above i in $\Gamma_{\bar{f}}$; in other words, $\mathfrak{v}_i = \lvert \mathrm{V}(\mathrm{Tree}_{\Gamma_{\bar{f}}}(i)) \rvert - 1$
V	17	variable denoting the vertex set of a digraph
$\mathrm{V}(\Gamma)$	9	the vertex set of Γ

$\mathcal{V}_{i,L}$	198	the arithmetic partition $\mathfrak{P}(\eta_{i,l}(x) : l \in \mathfrak{C}_{i,L})$ of C_i; if all f-periodic points in C_i have f-cycle length at most L, then $\mathcal{V}_{i,L}$ is the unique arithmetic partition of C_i such that one block of $\mathcal{V}_{i,L}$ consists of all f-transient points in C_i, while every other block of $\mathcal{V}_{i,L}$ consists of all f-periodic points of a common f-cycle length
w	21	variable denoting an element of the same ground set as x, y, z
$\mathfrak{w}$	63	variable denoting an edge weight
$\mathfrak{w}_k$	168	notation used in the proof of Lemma 5.3.2.2 (4); for fixed $i \in \{0, 1, \ldots, d-1\}$ and $t \in \mathbb{Z}$, it denotes the weight of the k-th edge from the left in the drawing of $\mathfrak{T}_{i_t,k}$ in Section 4.1
W	12	Lambert W function
$\mathcal{W}_{i,L}$	197	the arithmetic partition $\mathcal{V}_{i,L} \wedge \mathcal{Q}_{i,H_i+L-1}$ of C_i; if all f-periodic points in C_i have f-cycle length at most L, then for each f-periodic $x \in C_i$, the $\mathcal{W}_{i,L}$-block in which x is contained controls the isomorphism type of the connected component of Γ_f containing x
wreath product	8	none
x	1	an element of a ground set such as X or $\mathbb{F}_q$, or a variable in a congruence, depending on the context
x_n	3	a formal variable, used in cycle types
$x^{(t)}$	195	defined for $x \in \mathrm{per}(f)$ and $t \in \mathbb{Z}$; denotes $(f_{\lvert \mathrm{per}(f)})^t(x)$
$\mathfrak{x}$	9	variable denoting an element of $\mathfrak{X}$
$[\vec{\mathfrak{x}}]$	9	the cyclic sequence associated with the finite sequence $\vec{\mathfrak{x}}$ over $\mathfrak{X}$; also written $[\mathfrak{x}_0, \ldots, \mathfrak{x}_{L-1}]$ if $\vec{\mathfrak{x}} = (\mathfrak{x}_0, \ldots, \mathfrak{x}_{L-1})$
$\lvert\vec{\mathfrak{x}}\rangle$	90	defined when $\vec{\mathfrak{x}} \in \{0,1\}^n$; it is the n-qubit register encoding $\vec{\mathfrak{x}}$; physicists call this object a *ket*
X	1	a set (usually assumed to be finite)
$\mathcal{X}$	49	variable denoting an arithmetic partition; used, e.g., with enumerating indices $(\mathcal{X}_1, \mathcal{X}_2, \ldots)$ to avoid a clash with the notation $\mathcal{P}_i$
$\mathfrak{X}$	9	variable used to denote a set viewed as an alphabet, from whose elements finite sequences are formed
y	9	variable denoting an element of the same ground set as x
Y_i	117	the domain of definition of par_i, par'_i and par''_i
$Y_{i,p}$	118	the domain of definition of $\mathrm{par}'_{i,p}$
$\bar{Y}_i$	122	the parameter set $\prod_{p \in \mathfrak{P}_i} Y_{i,p}$

$\mathcal{Y}$	222	variable denoting an arithmetic partition (like $\mathcal{X}$)
$\mathfrak{Y}$	152	(used in Theorem 5.3.1.1) a special set of primes
z	15	variable denoting an element of the same ground set as x or y
$\mathcal{Z}_i$	87	appears in the definition of a partition-tree register; a specifically defined object that encodes enough information to reconstruct $\mathcal{P}_i$ (sometimes more)
$\mathbb{Z}/m\mathbb{Z}$	1	the ring of residues modulo m, with underlying set $\{0, 1, \ldots, m-1\}$
$(\mathbb{Z}/m\mathbb{Z})^*$	33	the multiplicative group of units of $\mathbb{Z}/m\mathbb{Z}$
α_i	47	the linear coefficient of A_i
$\bar{\alpha}_i$	53	the linear coefficient of $\mathcal{A}_i$
$\bar{\alpha}_{i,h}$	54	the linear coefficient of $\mathcal{A}_{i,h}$
$\bar{\alpha}'_i$	207	the linear coefficient of $\mathcal{A}'_i$
β_i	47	the constant coefficient of A_i
$\bar{\beta}_i$	118	the constant coefficient of $\mathcal{A}_i$
$\bar{\beta}_{i,h}$	54	the constant coefficient of $\mathcal{A}_{i,h}$
$\bar{\beta}'_i$	207	the constant coefficient of $\mathcal{A}'_i$
$\bar{\beta}''_i$	208	the constant coefficient of $(\mathcal{A}'_i)^{\mathrm{ord}_{s'_i}(\bar{\alpha}'_i)}$
γ	12	Euler–Mascheroni constant
Γ	1	variable denoting a (usually finite) digraph
$[\Gamma]_{\cong}$	18	the digraph isomorphism type of Γ
Γ^*	17	the dual digraph of Γ
Γ_g	1	the functional graph of g
$\Gamma_f^{(i)}$	83	the functional graph of $f_{\mid U_i}$
Γ_{per}	13	the induced subgraph of Γ_f on $\bigcup_{i \in \mathrm{per}(\bar{f})} C_i$
Δ	18	variable denoting a finite directed rooted tree
ζ	32	variable denoting a cycle of a function
$\eta_{i,l}(x)$	198	a certain s-congruence, which in case l is a multiple of ℓ (the $\bar{f}$-cycle length of i) characterizes the f-periodic points $x \in C_i$ with $f^l(x) = x$
$\theta_{i,h}(x)$	54	an s-congruence that characterizes when $x \in \mathbb{Z}/s\mathbb{Z}$ lies in the image of $\mathcal{A}_{i,h}$

$\Theta(x)$	55	a system of m-CCs in the single variable x
$\Theta_{i,h}(x)$	54	a system of (at most two) s-CCs that characterizes when $x \in C_i \cong \mathbb{Z}/s\mathbb{Z}$ has $\mathfrak{h}$-value h
ι	40	variable denoting an isomorphism or bijection; see in particular the notation $\iota_{\vec{r}}$ introduced after Problem 2.3.7
ι_i	8	the bijection $\mathbb{Z}/s\mathbb{Z} \to C_i$, $x \mapsto \omega^{i+dx}$
$\kappa_{i,p}$	120	shorthand for $\nu_p^{(\kappa_p)}(\bar{\beta}_i)$
κ_p	118	shorthand for $\nu_p(s)$
$\kappa_{\mathcal{P},A}(\vec{v},\vec{v}',J)$	28	technical parameter, used in the definition of $\sigma_{\mathcal{P},A}(\vec{v},\vec{v}')$
$\lambda(\mathcal{P},A)$	55	like $\mathfrak{P}'(\mathcal{P},A)$, with the last spanning congruence deleted
$\lambda_i^t(\mathcal{P})$	56	defined when $i < d$ is $\bar{f}$-periodic, $t \in \mathbb{N}_0$, and $\mathcal{P}$ is an arithmetic partition of C_i; it is the arithmetic partition of C_{i_t} defined recursively via $\lambda_i^0(\mathcal{P}) := \mathcal{P}$ and $\lambda_i^{t+1}(\mathcal{P}) := \lambda(\lambda_i^t(\mathcal{P}), A_{i_t})$
Λ	32	variable denoting a lift function $\mathbb{Z}/m_1\mathbb{Z} \to \mathbb{Z}/m_2\mathbb{Z}$, where $m_1, m_2 \in \mathbb{N}^+$ with $m_1 \mid m_2$ (i.e., $\Lambda(x) \equiv x \pmod{m_1}$ for all $x \in \mathbb{Z}/m_1\mathbb{Z}$); the details of its definition vary by context
μ_a	16	the endomorphism $x \mapsto ax$ of the group $\mathbb{Z}/m\mathbb{Z}$; m must be clear from context
ν	11	variable denoting a logical sign ($\emptyset$ or $\neg$)
$\nu_{l,l'}$	198	(used in Section 5.3.3) notation for a logical sign; it is defined as $\emptyset$ (the positive logical sign) if $l \mid l'$, and as $\neg$ otherwise
$\vec{\nu}_{i,L,l}$	198	the logical sign tuple $(\nu_{l,l'})_{l' \in \mathfrak{C}_{i,L}}$; if all f-periodic points in C_i have f-cycle length at most L and $l \in \mathfrak{C}_{i,L}$, then $\mathcal{B}(\mathcal{V}_{i,L}, \vec{\nu}_{i,L,l})$ consists precisely of those f-periodic $x \in C_i$ that have f-cycle length exactly l
$\vec{v}^+$	148	a special notation used in Section 5.2.3
$\vec{v}'^-$	148	a special notation used in Section 5.2.3; inverse to the notation $\vec{v}^+$
$\nu_p(n)$	16	the p-adic valuation of n
$\nu_p^{(v)}(n)$	16	defined as $\min\{v, \nu_p(n)\}$
$\vec{\xi}_{i,k}$	55	defined when $i \in \{0,1,\dots,d-1\}$ is $\bar{f}$-periodic and $k \in \{0,1,\dots,H_i\}$; it is the logical sign tuple of length H_i in which precisely the first k entries are equal to the positive logical sign; the block $\mathcal{B}(\mathcal{U}_i, \vec{\xi}_{i,k})$ consists just of those $x \in C_i$ that are of $\mathfrak{h}$-value k
$\pi(m)$	40	the set of prime divisors of m

$\rho_{\mathfrak{l}}$	23	the left-regular representation of a group G (clear from context) on itself, i.e., the function $G \to \mathrm{Sym}(G)$, $x \mapsto (y \mapsto xy)$
$\rho_{\mathfrak{r}}$	23	the right-regular representation of a group G (clear from context) on itself, i.e., the function $G \to \mathrm{Sym}(G)$, $x \mapsto (y \mapsto yx)$
σ	12	divisor sum function
$\sigma_{\mathcal{P},A}(\vec{\nu},\vec{\nu'})$	28	a technical parameter, for the significance of which see the comments on $\mathfrak{P}'(\mathcal{P},A)$
$\tau(m)$	156	the number of (positive) divisors of $m \in \mathbb{N}^+$
ϕ	33	Euler's totient function
φ	23	a variable denoting a group endomorphism
φ_i	47	defined for $i \in \{0,1,\ldots,d-1\}$ such that $a_i \neq 0$; it is μ_{α_i}, the group endomorphism of $\mathbb{Z}/s\mathbb{Z}$ associated with A_i
χ	90	a variable denoting a characteristic function
ψ	3	a permutation of X
ω	3	a primitive element of $\mathbb{F}_q$

Table A.2. Tabular overview of notation and terminology used in this memoir.

References

[1] L. Adleman, A subexponential algorithm for the discrete logarithm problem with applications to cryptography (abstract). In *20th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 55–60, IEEE, New York, 1979

[2] M. Agrawal, CS681: Computational number theory and algebra. Lecture 1 & 2: Integer and modular arithmetic. 2009, lecture notes, https://www.cse.iitk.ac.in/users/manindra/CS681/lecture1and2.pdf, visited on 2 September 2025

[3] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P. *Ann. of Math. (2)* **160** (2004), no. 2, 781–793

[4] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error rate. *SIAM J. Comput.* **38** (2008), no. 4, 1207–1282

[5] S. Ahmad, Cycle structure of automorphisms of finite cyclic groups. *J. Combinatorial Theory* **6** (1969), 370–374

[6] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The design and analysis of computer algorithms*. 2nd edn., Addison-Wesley Ser. Comput. Sci. Inf. Process., Addison-Wesley, MA, 1975

[7] F. Annexstein, M. Baumslag, and A. L. Rosenberg, Group action graphs and parallel architectures. *SIAM J. Comput.* **19** (1990), no. 3, 544–569

[8] T. M. Apostol, *Introduction to analytic number theory*. Undergrad. Texts Math., Springer, New York, 1976

[9] L. Babai, Graph isomorphism in quasipolynomial time. Version 2.5. 2018, preprint, https://people.cs.uchicago.edu/∼laci/quasi25.pdf, visited on 2 September 2025

[10] E. Bach, Comments on search procedures for primitive roots. *Math. Comp.* **66** (1997), no. 220, 1719–1727

[11] E. Bach and J. Shallit, *Algorithmic number theory. Vol. 1*. Found. Comput. Ser., MIT Press, Cambridge, MA, 1996

[12] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, Quantum lower bounds by polynomials. *J. ACM* **48** (2001), no. 4, 778–797

[13] A. Bors, On the dynamics of endomorphisms of finite groups. *Appl. Algebra Engrg. Comm. Comput.* **28** (2017), no. 3, 205–214

[14] A. Bors and Q. Wang, Coset-wise affine functions and cycle types of complete mappings. *Finite Fields Appl.* **83** (2022), article no. 102088

[15] A. Bors and Q. Wang, Generalized cyclotomic mappings: Switching between polynomial, cyclotomic, and wreath product form. *Commun. Math. Res.* **38** (2022), no. 2, 246–318

[16] A. Caranti, Quasi-inverse endomorphisms. *J. Group Theory* **16** (2013), no. 5, 779–792

[17] P.-Y. Chen, Solutions to introduction to algorithms third edition. 31-2 Analysis of bit operations in Euclid's algorithm. 2023, online resource, https://walkccc.me/CLRS/Chap31/Problems/31-2/, visited on 2 September 2025

[18] W.-S. Chou and I. E. Shparlinski, On the cycle structure of repeated exponentiation modulo a prime. *J. Number Theory* **107** (2004), no. 2, 345–356

[19] A. H. Clifford and G. B. Preston, *The algebraic theory of semigroups. Vol. II*. Math. Surveys 7, American Mathematical Society, Providence, RI, 1967

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. 3rd edn., MIT Press, Cambridge, MA, 2009

[21] G. Deng, Isomorphic digraphs from affine maps of finite cyclic groups. *ISRN Combinatorics* **2013** (2013), article no. 398641

[22] J. Dubrois and J.-G. Dumas, Efficient polynomial time algorithms computing industrial-strength primitive roots. *Inform. Process. Lett.* **97** (2006), no. 2, 41–45

[23] P. Flajolet and A. M. Odlyzko, Random mapping statistics. In *Advances in cryptology—EUROCRYPT '89 (Houthalen, 1989)*, edited by J. J. Quisquater and J. Vandewalle, pp. 329–354, Lecture Notes in Comput. Sci. 434, Springer, Berlin, 1990

[24] Z. Gao and Q. Wang, A probabilistic approach to value sets of polynomials over finite fields. *Finite Fields Appl.* **33** (2015), 160–174

[25] D. Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics. Proceedings of the American Mathematical Society Short Course held in Washington, DC, January 3–4, 2009*, edited by S. J. Lomonaco, Jr., pp. 13–58, Proc. Sympos. Appl. Math. 68, American Mathematical Society, Providence, RI, 2010

[26] L. K. Grover, A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, edited by G. L. Miller, pp. 212–219, ACM Press, New York, 1996

[27] H. Halberstam, On the distribution of additive number-theoretic functions. III. *J. London Math. Soc.* **31** (1956), 14–27

[28] H. Halberstam, Footnote to the Titchmarsh–Linnik divisor problem. *Proc. Amer. Math. Soc.* **18** (1967), 187–188

[29] D. Harvey and M. Hittmeir, A log-log speedup for exponent one-fifth deterministic integer factorisation. *Math. Comp.* **91** (2022), no. 335, 1367–1379

[30] D. Harvey and J. van der Hoeven, Integer multiplication in time $O(n \log n)$. *Ann. of Math. (2)* **193** (2021), no. 2, 563–617

[31] H. Helfgott, J. Bajpai, and D. Dona, Graph isomorphisms in quasi-polynomial time. 2017, arXiv:1710.04574v1

[32] H. A. Helfgott, Isomorphismes de graphes en temps quasi-polynomial [d'après Babai et Luks, Weisfeiler–Leman,...]. *Astérisque* **2019** (2019), no. 407, 135–182; Séminaire Bourbaki. Vol. 2016/2017. Exposés 1120–1135

[33] R. A. Hernández Toledo, Linear finite dynamical systems. *Comm. Algebra* **33** (2005), no. 9, 2977–2989

[34] W. M. L. Holcombe, *Algebraic automata theory*. Cambridge Stud. Adv. Math. 1, Cambridge University Press, Cambridge, 1982

[35] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation*. 3rd edn., Addison-Wesley, Boston, 2007

[36] A. S. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman, Reverse-engineering of polynomial dynamical systems. *Adv. in Appl. Math.* **39** (2007), no. 4, 477–489

[37] P. Kaye, R. Laflamme, and M. Mosca, *An introduction to quantum computing*. Oxford University Press, Oxford, 2007

[38] A. Y. Kitaev, Fault-tolerant quantum computation by anyons. *Ann. Physics* **303** (2003), no. 1, 2–30

[39] E. Knill, R. Laflamme, and W. H. Zurek, Resilient quantum computation. *Science* **279** (1998), no. 5349, 342–345

[40] D. E. Knuth, *The art of computer programming. Vol. 3. Sorting and searching*. 2nd edn., Addison-Wesley, Reading, MA, 1998

[41] R. Kumanduri and C. Romero, *Number theory with computer applications*. Prentice Hall, Upper Saddle River, NJ, 1998

[42] S. Lang, *Introduction to Diophantine approximations*. 2nd edn., Springer, New York, 1995

[43] R. Laubenbacher and B. Pareigis, Equivalence relations on finite dynamical systems. *Adv. in Appl. Math.* **26** (2001), no. 3, 237–251

[44] M. Le Borgne, A. Benveniste, and P. Le Guernic, Polynomial dynamical systems over finite fields. In *Algebraic computing in control. Proceedings of the First European Conference held in Paris, March 13–15, 1991*, edited by G. Jacob and F. Lamnabhi-Lagarrigue, pp. 212–222, Lect. Notes Control Inf. Sci. 165, Springer, Berlin, 1991

[45] H. W. Lenstra, Jr. and C. Pomerance, Primality testing with gaussian periods. 2011, preprint, https://math.dartmouth.edu/∼carlp/aks041411.pdf, visited on 2 September 2025

[46] J. V. Linnik, *The dispersion method in binary additive problems*. Transl. Math. monographs 4, American Mathematical Society, Providence, RI, 1963

[47] L. Lovász, *Combinatorial problems and exercises*. 2nd edn., North-Holland, Amsterdam, 1993

[48] M. Martelli, *Introduction to discrete dynamical systems and chaos*. Wiley-Intersci. Ser. Discrete Math. Optim., Wiley-Interscience, New York, 1999

[49] R. Martins, D. Panario, and C. Qureshi, A survey on iterations of mappings over finite fields. In *Combinatorics and finite fields—difference sets, polynomials, pseudorandomness and applications*, pp. 135–172, Radon Ser. Comput. Appl. Math. 23, De Gruyter, Berlin, 2019

[50] R. S. V. Martins and D. Panario, On the heuristic of approximating polynomials over finite fields by random mappings. *Int. J. Number Theory* **12** (2016), no. 7, 1987–2016

[51] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC Press Ser. Discrete Math. Appl., CRC Press, Boca Raton, FL, 1997

[52] G. A. Miller and H. C. Moreno, Non-abelian groups in which every subgroup is abelian. *Trans. Amer. Math. Soc.* **4** (1903), no. 4, 398–404

[53] J. C. P. Miller, On factorisation, with a suggested new approach. *Math. Comp.* **29** (1975), 155–172

[54] D. Milligan and M. Wilson, The behaviour of affine Boolean sequential networks. *Connection Science* **5** (1993), no. 2, 153–167

[55] H. L. Montgomery and R. C. Vaughan, The large sieve. *Mathematika* **20** (1973), 119–134

[56] H. Niederreiter and A. Winterhof, Cyclotomic $\mathcal{R}$-orthomorphisms of finite fields. *Discrete Math.* **295** (2005), no. 1-3, 161–171

[57] D. Panario and L. Reis, The functional graph of linear maps over finite fields and applications. *Des. Codes Cryptogr.* **87** (2019), no. 2-3, 437–453

[58] A. Peinado, F. Montoya, J. Muñoz, and A. J. Yuste, Maximal periods of $x^2 + c$ in $\mathbb{F}_q$. In *Applied algebra, algebraic algorithms and error-correcting codes. Proceedings of the 14th International Symposium (AAECC-14) held in Melbourne, November 26–30, 2001*, edited by S. Boztaş and I. E. Shparlinski, pp. 219–228, Lecture Notes in Comput. Sci. 2227, Springer, Berlin, 2001

[59] J. M. Pollard, A Monte Carlo method for factorization. *Nordisk Tidskr. Informationsbehandling (BIT)* **15** (1975), no. 3, 331–334

[60] C. Pomerance, Fast, rigorous factorization and discrete logarithm algorithms. In *Discrete algorithms and complexity. Proceedings of the Japan-U.S. joint seminar held in Kyoto, June 4–6, 1986*, edited by D. S. Johnson et al., pp. 119–143, Perspect. Comput. 15, Academic Press, Boston, MA, 1987

[61] C. Qureshi and D. Panario, Rédei actions on finite fields and multiplication map in cyclic group. *SIAM J. Discrete Math.* **29** (2015), no. 3, 1486–1503

[62] C. Qureshi and D. Panario, The graph structure of Chebyshev polynomials over finite fields and applications. *Des. Codes Cryptogr.* **87** (2019), no. 2-3, 393–416

[63] G. Robin, Grandes valeurs de la fonction somme des diviseurs et hypothèse de Riemann. *J. Math. Pures Appl. (9)* **63** (1984), no. 2, 187–213

[64] A. Schönhage, Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Inform.* **1** (1971), 139–144

[65] A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)* **7** (1971), 281–292

[66] M. Sha, Digraphs from endomorphisms of finite cyclic groups. *J. Combin. Math. Combin. Comput.* **83** (2012), 105–120

[67] C. E. Shannon, The synthesis of two-terminal switching circuits. *Bell System Tech. J.* **28** (1949), 59–98

[68] P. W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science. Proceedings of the IEEE Symposium held in Santa Fe, NM, November 20–22, 1994*, edited by S. Goldwasser, pp. 124–134, IEEE Computer Society Press, Los Alamitos, CA, 1994

[69] P. W. Shor, Fault-tolerant quantum computation. In *37th Annual Symposium on Foundations of Computer Science. Held in Burlington, VT, October 14–16, 1996*, pp. 56–65, IEEE Computer Society Press, Los Alamitos, CA, 1996

[70] The GAP Group, GAP–groups, algorithms, and programming. 2022, version 4.12.0, http://www.gap-system.org, visited on 2 September 2025

[71] S. Ugolini, Graphs associated with the map $x \mapsto x + x^{-1}$ in finite fields of characteristic two. In *Theory and applications of finite fields. Proceedings of the 10th International Conference on Finite Fields and Their Applications (Fq 10) held in Ghent, July 11–15, 2011*, edited by M. Lavrauw et al., pp. 187–204, Contemp. Math. 579, American Mathematical Society, Providence, RI, 2012

[72] S. Ugolini, Graphs associated with the map $X \mapsto X + X^{-1}$ in finite fields of characteristic three and five. *J. Number Theory* **133** (2013), no. 4, 1207–1228

[73] T. Vasiga and J. Shallit, On the iteration of certain quadratic maps over GF(p). *Discrete Math.* **277** (2004), no. 1-3, 219–240

[74] A. Veliz-Cuba and R. Laubenbacher, On the computation of fixed points in Boolean networks. *J. Appl. Math. Comput.* **39** (2012), no. 1-2, 145–153

[75] J. von zur Gathen and J. Gerhard, *Modern computer algebra*. 3rd edn., Cambridge University Press, Cambridge, 2013

[76] J. von zur Gathen and D. Panario, Factoring polynomials over finite fields: A survey. *J. Symbolic Comput.* **31** (2001), no. 1-2, 3–17

[77] S. Wagstaff, The Cunningham project. 2022, online database, https://homes.cerias.purdue.edu/∼ssw/cun/, visited on 2 September 2025

[78] D. Q. Wan and R. Lidl, Permutation polynomials of the form $x^r f(x^{(q-1)/d})$ and their group structure. *Monatsh. Math.* **112** (1991), no. 2, 149–163

[79] Q. Wang, Cyclotomic mapping permutation polynomials over finite fields. In *Sequences, subsequences, and consequences. Revised invited papers from the International Workshop (SSC 2007) held at the University of Southern California, Los Angeles, CA, May 31–June 2, 2007*, edited by S. W. Golomb et al., pp. 119–128, Lecture Notes in Comput. Sci. 4893, Springer, Berlin, 2007

[80] Q. Wang, Cyclotomy and permutation polynomials of large indices. *Finite Fields Appl.* **22** (2013), 57–69

[81] Q. Wang, A note on inverses of cyclotomic mapping permutation polynomials over finite fields. *Finite Fields Appl.* **45** (2017), 422–427

[82] Q. Wang, Polynomials over finite fields: An index approach. In *Combinatorics and finite fields—difference sets, polynomials, pseudorandomness and applications*, edited by K.-U. Schmidt and A. Winterhof, pp. 319–346, Radon Ser. Comput. Appl. Math. 23, De Gruyter, Berlin, 2019

[83] X. Wang and V. Y. Pan, Acceleration of Euclidean algorithm and rational number reconstruction. *SIAM J. Comput.* **32** (2003), no. 2, 548–556

[84] J. Watrous, Quantum computational complexity. In *Computational complexity. Vols. 1–6*, pp. 2361–2387, Springer, New York, 2012

[85] W.-D. Wei, X.-H. Gao, and B.-F. Yang, Equivalence relation on the set of subsets of z_v and enumeration of the equivalence classes (Research Announcement). *Adv. Math.* **17** (1988), 326–327

[86] W. D. Wei and J. Y. Xu, Cycle index of direct product of permutation groups and number of equivalence classes of subsets of Z_v. *Discrete Math.* **123** (1993), no. 1-3, 179–188

[87] Y. Zheng, Y. Yu, Y. Zhang, and D. Pei, Piecewise constructions of inverses of cyclotomic mapping permutation polynomials. *Finite Fields Appl.* **40** (2016), 1–9

MEMOIRS OF THE EUROPEAN MATHEMATICAL SOCIETY

Alexander Bors, Daniel Panario, Qiang Wang

Functional Graphs of Generalized Cyclotomic Mappings of Finite Fields

The functional graph of a function $g : X \to X$ is the directed graph with vertex set X the edges of which are of the form $x \to g(x)$ for $x \in X$. Functional graphs are studied because they allow one to understand the behavior of g under iteration (i.e., to understand the discrete dynamical system (X, g)), which has various applications, especially when X is a finite field $\mathbb{F}_q$. This memoir is an extensive study of the functional graphs of so-called index d generalized cyclotomic mappings of $\mathbb{F}_q$, which are a natural and manageable generalization of monomial functions. We provide both theoretical results on the structure of their functional graphs and Las Vegas algorithms for solving fundamental problems, such as parametrizing the connected components of the functional graph by representative vertices, or describing the structure of a connected component given by a representative vertex. The complexity of these algorithms is analyzed in detail, and we make the point that for fixed index d and most prime powers q (in the sense of asymptotic density), suitable implementations of these algorithms have an expected runtime that is polynomial in $\log q$ on quantum computers, whereas their expected runtime is subexponential in $\log q$ on a classical computer. We also discuss four special cases in which one can devise Las Vegas algorithms with this kind of complexity behavior over most finite fields that solve the graph isomorphism problem for functional graphs of generalized cyclotomic mappings.

https://ems.press
ISSN 2747-9080
ISBN 978-3-98547-101-0

EMS PRESS