

**Mathematical
Surveys
and
Monographs**

Volume 177



Non-commutative Cryptography and Complexity of Group-theoretic Problems

**Alexei Myasnikov
Vladimir Shpilrain
Alexander Ushakov**

*With an appendix by
Natalia Mosina*



American Mathematical Society

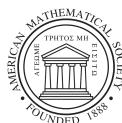
**Mathematical
Surveys
and
Monographs**

Volume 177

Non-commutative Cryptography and Complexity of Group-theoretic Problems

Alexei Myasnikov
Vladimir Shpilrain
Alexander Ushakov

With an appendix by
Natalia Mosina



American Mathematical Society
Providence, Rhode Island

EDITORIAL COMMITTEE

Ralph L. Cohen, Chair Michael A. Singer
Jordan S. Ellenberg Benjamin Sudakov
Michael I. Weinstein

2010 *Mathematics Subject Classification.* Primary 94A60, 20F10, 68Q25, 94A62, 11T71.

For additional information and updates on this book, visit
www.ams.org/bookpages/surv-177

Library of Congress Cataloging-in-Publication Data

Myasnikov, Alexei G., 1955–

Non-commutative cryptography and complexity of group-theoretic problems / Alexei Myasnikov, Vladimir Shpilrain, Alexander Ushakov ; with an appendix by Natalia Mosina.

p. cm. – (Mathematical surveys and monographs ; v. 177)

Includes bibliographical references and index.

ISBN 978-0-8218-5360-3 (alk. paper)

1. Combinatorial group theory. 2. Cryptography. 3. Computer algorithms. 4. Number theory.

I. Shpilrain, Vladimir, 1960– II. Ushakov, Alexander. III. Title.

QA182.5.M934 2011

005.8'2–dc23

2011020554

Copying and reprinting. Individual readers of this publication, and nonprofit libraries acting for them, are permitted to make fair use of the material, such as to copy a chapter for use in teaching or research. Permission is granted to quote brief passages from this publication in reviews, provided the customary acknowledgment of the source is given.

Republication, systematic copying, or multiple reproduction of any material in this publication is permitted only under license from the American Mathematical Society. Requests for such permission should be addressed to the Acquisitions Department, American Mathematical Society, 201 Charles Street, Providence, Rhode Island 02904-2294 USA. Requests can also be made by e-mail to reprint-permission@ams.org.

© 2011 by the American Mathematical Society. All rights reserved.

The American Mathematical Society retains all rights except those granted to the United States Government.

Printed in the United States of America.

- ∞ The paper used in this book is acid-free and falls within the guidelines established to ensure permanence and durability.
Visit the AMS home page at <http://www.ams.org/>

To our children: Nikita, Olga, Marie, . . .

Contents

Preface	xiii
Introduction	1
Part 1. Background on Groups, Complexity, and Cryptography	
Chapter 1. Background on Public-Key Cryptography	7
1.1. From key establishment to encryption	8
1.2. The Diffie-Hellman key establishment	8
1.3. The ElGamal cryptosystem	9
1.4. The RSA cryptosystem	10
1.5. Rabin's cryptosystem	11
1.6. Authentication	11
1.6.1. The Feige-Fiat-Shamir scheme	12
Chapter 2. Background on Combinatorial Group Theory	13
2.1. Basic definitions and notation	13
2.2. Presentations of groups by generators and relators	14
2.3. Algorithmic problems of group theory: decision, witness, search	15
2.3.1. The word problem	15
2.3.2. The conjugacy problem	16
2.3.3. The decomposition and factorization problems	16
2.3.4. The membership problem	17
2.3.5. The isomorphism problem	17
2.3.6. More on search/witness problems	18
2.4. Nielsen's and Schreier's methods	20
2.5. Tietze's method	21
2.6. Normal forms	22
Chapter 3. Background on Computational Complexity	25
3.1. Algorithms	25
3.1.1. Deterministic Turing machines	25
3.1.2. Non-deterministic Turing machines	26
3.1.3. Probabilistic Turing machines	26
3.2. Computational problems	26
3.2.1. Decision and search computational problems	27
3.2.2. Size functions	28

3.2.3.	Stratification	30
3.2.4.	Reductions and complete problems	31
3.2.5.	Many-to-one reductions	31
3.2.6.	Turing reductions	31
3.3.	The worst case complexity	32
3.3.1.	Complexity classes	32
3.3.2.	Class NP	33
3.3.3.	Polynomial-time many-to-one reductions and class NP	34
3.3.4.	NP -complete problems	35
3.3.5.	Deficiency of the worst case complexity	37
Part 2. Non-commutative Cryptography		
Chapter 4.	Canonical Non-commutative Cryptography	41
4.1.	Protocols based on the conjugacy search problem	41
4.2.	Protocols based on the decomposition problem	43
4.2.1.	“Twisted” protocol	44
4.2.2.	Hiding one of the subgroups	44
4.2.3.	Commutative subgroups	45
4.2.4.	Using matrices	45
4.2.5.	Using the triple decomposition problem	45
4.3.	A protocol based on the factorization search problem	46
4.4.	Stickel’s key exchange protocol	47
4.4.1.	Linear algebra attack	48
4.5.	The Anshel-Anshel-Goldfeld protocol	50
4.6.	Relations between different problems	51
Chapter 5.	Platform Groups	55
5.1.	Braid groups	55
5.1.1.	A group of braids and its presentation	56
5.1.2.	Dehornoy handle free form	58
5.1.3.	Garside normal form	59
5.2.	Thompson’s group	60
5.3.	Groups of matrices	63
5.4.	Small cancellation groups	65
5.4.1.	Dehn’s algorithm	65
5.5.	Solvable groups	65
5.5.1.	Normal forms in free metabelian groups	66
5.6.	Artin groups	68
5.7.	Grigorchuk’s group	69
Chapter 6.	More Protocols	73
6.1.	Using the subgroup membership search problem	73
6.1.1.	Groups of the form $F/[R, R]$	76
6.2.	The MOR cryptosystem	76

Chapter 7. Using Decision Problems in Public-Key Cryptography	79
7.1. The Shpilrain-Zapata scheme	79
7.1.1. The protocol	80
7.1.2. Pool of group presentations	82
7.1.3. Generating random elements in finitely presented groups	83
7.2. Public-key encryption and encryption emulation attacks	85
Chapter 8. Authentication	91
8.1. A Diffie-Hellman-like scheme	91
8.2. A Feige-Fiat-Shamir-like scheme	92
8.3. Authentication based on the twisted conjugacy problem	93
8.4. Authentication from matrix conjugation	94
8.4.1. The protocol, beta version	94
8.4.2. The protocol, full version	95
8.4.3. Cryptanalysis	96
8.5. Authentication from actions on graphs, groups, or rings	97
8.5.1. When a composition of functions is hard-to-invert	97
8.5.2. Three protocols	98
8.5.3. Subgraph isomorphism	100
8.5.4. Graph homomorphism	101
8.5.5. Graph colorability	102
8.5.6. Endomorphisms of groups or rings	103
8.5.7. Platform: free metabelian group of rank 2	104
8.5.8. Platform: \mathbb{Z}_p^*	104
8.6. No-leak authentication by the Sherlock Holmes method	104
8.6.1. No-leak vs. zero-knowledge	105
8.6.2. The meta-protocol for authentication	106
8.6.3. Correctness of the protocol	107
8.6.4. Questions and answers	108
8.6.5. Why is the Graph ISO proof system not “no-leak”?	109
8.6.6. A particular realization: subset sum	109
8.6.7. A particular realization: polynomial equations	111

Part 3. Generic Complexity and Cryptanalysis

Chapter 9. Distributional Problems and the Average Case Complexity	117
9.1. Distributional computational problems	117
9.1.1. Distributions and computational problems	117
9.1.2. Stratified problems with ensembles of distributions	119
9.1.3. Randomized many-to-one reductions	119
9.2. Average case complexity	120
9.2.1. Polynomial on average functions	121
9.2.2. Average case behavior of functions	125
9.2.3. Average case complexity of algorithms	125

9.2.4. Average case vs. worst case	126
9.2.5. Average case behavior as a trade-off	126
9.2.6. Deficiency of the average case complexity	130
Chapter 10. Generic Case Complexity	131
10.1. Generic Complexity	131
10.1.1. Generic sets	131
10.1.2. Asymptotic density	132
10.1.3. Convergence rates	133
10.1.4. Generic complexity of algorithms and algorithmic problems	134
10.1.5. Deficiency of the generic complexity	135
10.2. Generic versus average case complexity	136
10.2.1. Comparing generic and average case complexities	136
10.2.2. When average polynomial time implies generic	136
10.2.3. When generically easy implies easy on average	138
Chapter 11. Generic Complexity of NP-complete Problems	141
11.1. The linear generic time complexity of subset sum problem	141
11.2. A practical algorithm for the subset sum problem	142
11.3. 3-Satisfiability	143
Chapter 12. Generic Complexity of Undecidable Problems	147
12.1. The halting problem	147
12.2. The Post correspondence problem	150
12.3. Finitely presented semigroups with undecidable word problem	150
Chapter 13. Strongly, Super, and Absolutely Undecidable Problems	155
13.1. Halting problem for Turing machines	157
13.2. Strongly undecidable problems	158
13.2.1. The halting problem is strongly undecidable	158
13.2.2. Strongly undecidable analog of Rice's theorem	159
13.3. Generic amplification of undecidable problems	160
13.4. Semigroups with super-undecidable word problems	163
13.5. Absolutely undecidable problems	165
Part 4. Asymptotically Dominant Properties and Cryptanalysis	
Chapter 14. Asymptotically Dominant Properties	171
14.1. A brief description	171
14.2. Random subgroups and generating tuples	172
14.3. Asymptotic properties of subgroups	173
14.4. Groups with generic free basis property	174
14.5. Quasi-isometrically embedded subgroups	176
Chapter 15. Length Based and Quotient Attacks	179

15.1.	Anshel-Anshel-Goldfeld scheme	179
15.1.1.	Description of the Anshel-Anshel-Goldfeld scheme	179
15.1.2.	Security assumptions of the AAG scheme	179
15.2.	Length based attacks	181
15.2.1.	A general description	181
15.2.2.	LBA in free groups	184
15.2.3.	LBA in groups from \mathcal{FB}_{exp}	185
15.3.	Computing the geodesic length in a subgroup	186
15.3.1.	Related algorithmic problems	186
15.3.2.	Geodesic length in braid groups	188
15.4.	Quotient attacks	189
15.4.1.	Membership problems in free groups	190
15.4.2.	Conjugacy problems in free groups	191
15.4.3.	The MSP and SCSP* in groups with “good” quotients	194

Part 5. Word and Conjugacy Search Problems in Groups

Chapter 16.	Word Search Problem	199
16.1.	Introduction	199
16.2.	Presentations of groups	203
16.3.	Approximating Cayley graphs of finitely presented groups	204
16.3.1.	Cayley graph approximations and singular subcomplexes	204
16.3.2.	van Kampen diagrams	208
16.3.3.	Depth of diagrams and the canonical embeddings	209
16.4.	New algorithms for the word search problem in groups	212
16.4.1.	Search problems in groups	212
16.4.2.	The word search problem in groups. Algorithm \mathcal{A} .	213
16.4.3.	The word search problem in groups. Algorithm \mathcal{B} .	215
16.5.	Random van Kampen diagrams	218
16.5.1.	Basic random extensions and simple random walks	218
16.5.2.	Probability and asymptotic measure on diagrams	219
16.5.3.	Iterative random generator RG_n	222
16.5.4.	Diagram complexity and random generator RG_χ	222
16.6.	Basic extension algorithm B_S and relative probability measures	224
16.6.1.	Basic extension B_S	224
16.6.2.	Completeness of the basic extension B_S	226
16.6.3.	Some properties of B_S	236
16.7.	Asymptotic properties of diagrams	237
16.7.1.	Properties related to RG_χ	237
16.8.	Generic properties of trivial words	242
16.8.1.	Random trivial words	242
16.8.2.	Generic properties of trivial words	243
16.9.	Comparison with standard techniques	245
16.9.1.	The Todd-Coxeter algorithm	245
16.9.2.	Total enumeration of $gp_F(R)$	246

Chapter 17. Conjugacy Search Problem	247
17.1. Introduction	247
17.2. Weighted graphs	248
17.2.1. Definition	248
17.2.2. Conjugacy and pseudo conjugacy graphs	249
17.3. Transformations of weighted X -digraphs	250
17.3.1. Shift operator	251
17.3.2. Stallings' fold	252
17.3.3. Stallings' procedure	254
17.3.4. Basic extension	256
17.3.5. R -extension algorithm	258
17.4. Conjugacy graphs	259
17.4.1. Existence of conjugacy graphs	260
17.4.2. Conjugacy graph approximation	261
17.5. Annular (Schupp) diagrams	262
17.5.1. Depth of annular diagrams	263
17.5.2. Annular diagrams as pseudo conjugacy graphs	264
17.6. The conjugacy search problem	264
17.6.1. Annular diagram bisection	264
17.6.2. Conjugacy search algorithm	269
17.7. Random annular diagrams	272
17.7.1. Basic random extension of annular diagrams	274
17.7.2. Completeness of B_S	278
17.8. Asymptotic properties of random annular diagrams	279
17.9. Asymptotic properties of conjugated words	280
17.9.1. Random conjugated words	280
17.9.2. Generic properties of random conjugated words	281

Part 6. Word Problem in some Special Classes of Groups

Chapter 18. Free Solvable Groups	285
18.1. Preliminaries	289
18.1.1. The word problem	289
18.1.2. Free solvable groups and the Magnus embedding	289
18.1.3. Free Fox derivatives	291
18.1.4. Flows on F/N	292
18.1.5. Geometric interpretation of Fox derivatives	294
18.1.6. Geometric circulations and the first homology group of Γ	296
18.1.7. Geodesics in F/N'	296
18.2. The word problem in free solvable groups	298
18.2.1. The word problem in free metabelian groups	298
18.2.2. The word problem in free solvable groups	300
18.3. Geodesics in free metabelian groups	303
18.3.1. Algorithmic problems about geodesics in groups	303
18.3.2. Reduction to M_2	304

18.3.3.	Rectilinear Steiner tree problem	305
18.3.4.	NP-completeness of BGLP in M_2	305
Chapter 19.	Compressed Words	309
19.1.	Straight line programs	309
19.2.	Basic operations over SLPs	310
19.2.1.	Subwords	310
19.2.2.	Inversion	312
19.3.	Plandowski's algorithm	312
19.3.1.	Assertions	312
19.3.2.	Trimming	314
19.3.3.	Splitting	315
19.3.4.	Compactification	316
19.3.5.	Satisfiability of SLP	318
19.3.6.	Plandowski's algorithm	319
19.4.	Longest common initial segment	319
19.5.	Reduction	320
19.6.	The word problem in the automorphism group of a free group	321
Appendix A.	Probabilistic Group-based Cryptanalysis	325
A.1.	Introduction	325
A.2.	Probability theory on groups	328
A.2.1.	Probabilities for various spaces – overview	328
A.2.2.	Mean (expectation) of a group-valued random element	331
A.2.3.	The mean set in a group	333
A.2.4.	Other possible definitions of \mathbb{E}	334
A.3.	Strong law of large numbers	334
A.3.1.	Separation and inclusion lemmas	336
A.3.2.	Case of multi-vertex mean-sets	339
A.4.	Concentration of measure inequalities	345
A.4.1.	Chebyshev's inequality for graphs/groups	345
A.4.2.	Chernoff-Hoeffding-like bound for graphs/groups	348
A.5.	Configurations of mean-sets	349
A.6.	Computation of mean-sets in graphs	352
A.6.1.	Experiments	353
A.7.	Probabilistic cryptanalysis of Sibert type protocols	354
A.7.1.	Outline	355
A.7.2.	Zero-knowledge interactive proof systems	355
A.7.3.	Description of the protocol	356
A.7.4.	Security of the protocol	357
A.7.5.	The idea of mean-set attack: the shift search problem	358
A.7.6.	Effective computation of a mean-set	359
A.7.7.	The mean-set attack	360
A.7.8.	Experiments	363
A.7.9.	Defending against the attack	366

Bibliography	369
Abbreviations and Notation	381
Index	383

Preface

In this book, we explore “non-commutative ideas” in cryptography that have appeared in the literature over the last decade or so. Since all three authors have backgrounds in combinatorial and computational group theory, we pay particular attention to what can be called group-based cryptography, i.e., cryptography that uses non-commutative group theory one way or another. However, we also discuss some ideas from other areas of mathematics, especially when we address the problem of authentication, which is one of the most important areas of applications of modern cryptography.

We also show that there is remarkable feedback from cryptography to combinatorial and computational group theory because some of the problems motivated by cryptography appear to be new to group theory, and they open many interesting research avenues within group theory. Then, we employ complexity theory, notably *generic-case complexity* of algorithms, for cryptanalysis of various cryptographic protocols based on infinite groups. We also use the ideas and machinery from the theory of generic-case complexity to study *asymptotically dominant properties* of some infinite groups that have been used in public-key cryptography so far. It turns out that for a relevant cryptographic scheme to be secure, it is essential that keys are selected from a “very small” (relative to the whole group, say) subset rather than from the whole group. Detecting these subsets (“black holes”) for a particular cryptographic scheme is usually a very challenging problem, but it holds the key to creating secure cryptographic primitives based on non-commutative groups.

A substantial part of the book deals with new directions in computational group theory itself, notably with search problems motivated by cryptography. We also study complexity of more traditional decision problems (like the word and conjugacy problems) in some groups that have been suggested as platforms for cryptographic protocols.

Acknowledgments. It is a pleasure for us to thank our colleagues who have directly or indirectly contributed to this book. In particular, we would like to thank M. Anshel, G. Baumslag, Y. Bryukhov, M. Elder, N. Fazio, B. Fine, R. Gilman, D. Grigoriev, Yu. Gurevich, G. Havas, D. Kahrobaei, I. Kapovich, L. Makar-Limanov, A. D. Miasnikov, A. A. Mikhalev, A. V. Mikhalev, D. Osin, G. Rosenberg, T. Riley, V. Roman’kov, A. Rybalov, M. Sapir, W. Skeith, R. Steinwandt, B. Tsaban, A. Vershik, G. Zapata for numerous helpful comments and insightful discussions.

We are also grateful to our home institutions, the City College of New York and Stevens Institute of Technology for a stimulating research environment. A. G. Myasnikov and A. Ushakov acknowledge support by the NSF grant DMS-0914773 during their work on this book. A. G. Myasnikov was also supported by an NSERC grant. V. Shpilrain acknowledges support by the NSF grant DMS-0914778.

Finally, we are indebted to S. I. Gelfand for his encouragement and patience during our work on this project, and to C. Thivierge and L. Cole for their help during the production process.

Alexei Myasnikov
Vladimir Shpilrain
Alexander Ushakov

New York

Introduction

The object of this book is threefold. First, we explore how non-commutative groups which are typically studied in combinatorial group theory can be used in *public-key cryptography*. Second, we show that there is a remarkable feedback from cryptography to combinatorial group theory because some of the problems motivated by cryptography appear to be new to group theory, and they open many interesting research avenues within group theory. In particular, we put a lot of emphasis on studying *search problems*, as compared to *decision problems*, traditionally studied in combinatorial group theory. Yet another purpose of this book is to survey recent developments in combinatorial and computational group theory that are or can potentially be useful to cryptography. This includes generic properties of groups, subgroups and elements, generic- and average-case complexity of group-theoretic algorithms, asymptotically dominant properties, compressed words, etc.

We emphasize that our focus in this book is on public-key (or asymmetric) cryptography. “Classical” (or symmetric) cryptography generally uses a single key which allows both for the encryption and decryption of messages. This form of cryptography is usually referred to as symmetric key cryptography because the same algorithm or procedure or key is used not only to encode a message but also to decode that message. The key being used then is necessarily private and known only to the parties involved in communication. This method for transmission of messages was basically the only way until 1976 when W. Diffie and M. Hellman introduced an ingenious new way of transmitting information, which has led to what is now known as public-key cryptography. The basic idea is quite simple. It involves the use of a so-called one-way function f to encrypt messages. Very informally, a one-way function f is a function such that it is easy to compute the value of $f(x)$ for each argument x in the domain of f , but it is very hard to compute the value of $f^{-1}(y)$ for “most” y in the range of f . The most celebrated one-way function, due to Rivest, Shamir and Adleman, gives rise to the protocol called RSA, which is the most common public-key cryptosystem in use today. It is employed, for instance, in the browsers Firefox and Internet Explorer. Thus it plays a critical and increasingly important role in all manner of secure electronic communication and transactions that use the Internet. It depends in its efficacy, as do many of other cryptosystems, on the complexity of finite abelian (or commutative) groups. Such algebraic structures are very special examples of *finitely generated groups*. Finitely generated groups have been intensively studied for over 150 years and they exhibit extraordinary complexity. Although the security of the Internet does not appear to be threatened at this time because of the weaknesses of the existing protocols such as RSA, it seems prudent to explore possible enhancements and replacements

of such protocols which depend on finite abelian groups. This is one of the basic objectives of this book.

The idea of using the complexity of infinite non-abelian groups in cryptography goes back to Wagner and Magyarik [180] who in 1985 devised a public-key protocol based on the unsolvability of the word problem for finitely presented groups (or so they thought). Their protocol now looks somewhat naive, but it was pioneering. More recently, there has been an increased interest in applications of non-abelian group theory to cryptography (see for example [5, 161, 257]). Most suggested protocols are based on search problems that grew out of more traditional *decision problems* of combinatorial group theory. Protocols based on search problems fit in with the general paradigm of a public-key protocol based on a one-way function. We therefore dub the relevant area of cryptography *canonical cryptography* and explore it in Chapter 4 of our book.

On the other hand, employing decision problems in public-key cryptography allows one to depart from the canonical paradigm and construct cryptographic protocols with new properties, impossible in the canonical model. In particular, such protocols can be secure against some “brute force” attacks by computationally unbounded adversary. There is a price to pay for that, but the price is reasonable: a legitimate receiver decrypts correctly with probability that can be made very close to 1, but not equal to 1. We discuss this and some other new ideas in Chapter 15.

In Chapter 8, we describe several new ideas in public-key authentication. The problem of secure public-key authentication is, arguably, the most intriguing one in public-key cryptography since it can be put “between” the two principal problems: (1) the existence of one-way functions; (2) the existence of a secure public-key cryptosystem (or, equivalently, the existence of a secure key exchange protocol). Namely, the existence of a secure public-key cryptosystem obviously implies the existence of a secure public-key authentication scheme, and the latter implies the existence of one-way functions. However, the reverse implications may not hold. We describe in this book a number of authentication schemes based on different ideas, some of them coming from group theory and some from other areas of mathematics. Independently interesting is a related concept of a “zero-knowledge” proof that we also discuss in Chapter 8.

There were attempts, so far rather isolated, to provide a rigorous mathematical justification of security for protocols based on infinite groups, as an alternative to the security model known as *semantic security* [102], which is widely accepted in the “finite case”. It turns out, not surprisingly, that to introduce such a model one would need to define a suitable probability measure on a given infinite group. This delicate problem has been addressed in [29, 28, 167] for some classes of groups, but this is just the beginning of the work required to build a solid mathematical foundation for assessing security of cryptosystems based on infinite groups. Another, related, area of research studies *generic* behavior of infinite groups with respect to various properties (see [146] and its references). It is becoming clear now that, as far as security of a cryptographic protocol is concerned, the appropriate measure of computational hardness of a group-theoretic problem in the core of such a cryptographic protocol should take into account the “generic” case of the problem, as opposed to the worst case or average case traditionally studied in mathematics and theoretical computer science. Generic-case performance of various algorithms on

groups has been studied in [146, 148], [149], and many other papers. It is the focus of Part 3 of this book.

We have to make a disclaimer though that we do *not* address here security properties (e.g. semantic security) that are typically considered in “traditional” cryptography. They are extensively treated in cryptographic literature; here we single out a forthcoming monograph [104] because it also studies how group theory may be used in cryptography, but the focus there is quite different from ours; in particular, the authors of [104] do not consider infinite groups, but they do study “traditional” security properties thoroughly. To avoid misunderstanding, we stress that we do not mean any disrespect for semantic security; it is just that in the present book, we place strong emphasis on discussing new ideas and the intuition behind them. One other thing to keep in mind is that here we deal mostly with *infinite* groups as platforms for cryptographic primitives, and possible ramifications and generalizations of semantic security to infinite platforms are yet to be explored, although we do make some inroads into that area in this book.

We also have to point out that, in comparison with some other books on cryptography, our book is less implementation-oriented. In particular, we usually avoid giving specific parameters for cryptographic protocols that we describe; instead, we focus on new ideas and new research avenues.

In Part 4 of our book, we use the ideas and machinery from Part 3 to study *asymptotically dominant properties* of some infinite groups that have been used in public-key cryptography so far. Informally, the point is that “most” elements, or tuples of elements, or subgroups, or whatever, of a given group have some “smooth” properties which make them misfits for being used (as private or public keys, say) in a cryptographic scheme. Therefore, for a relevant cryptographic scheme to be secure, it is essential that keys are actually sampled (i.e., randomly selected) from a “small” (relative to the whole group, say) subset rather than from the whole group. Detecting these subsets (“black holes”) for a particular cryptographic scheme is usually a very challenging problem, but it holds the key to creating secure cryptographic primitives based on infinite non-abelian groups.

Parts 5 and 6 are not about cryptography *per se*, but about complexity of various algorithmic problems in combinatorial group theory, notably of search problems, motivated by cryptography. Part 5 offers an in-depth study, from a computational perspective, of two major search problems in group theory: the word search and the conjugacy search problems. Chapter 18 of Part 6 describes new interesting developments in the algorithmic theory of solvable, in particular metabelian, groups. It may come as a surprise that in a free metabelian group where standard algorithmic problems are efficiently solvable, the *bounded geodesic problem* (i.e., deciding whether for a given word and a given integer k , there is another word of length $\leq k$ representing the same element) is **NP**-complete. Chapter 19 describes yet another spectacular new development related to complexity of group-theoretic problems. Based on the ideas of *compressed words* and straight-line programs coming from computer science, it is possible to design polynomial-time algorithms for some problems in group theory that were previously thought to be computationally intractable including, for example, the word problem in the automorphism group of a free group.

In the Appendix, yet another probabilistic approach to cryptanalysis is introduced. Specifically, a particular example of a group-based authentication protocol,

due to Sibert et al. [256], is used to illustrate how the “strong law of large numbers” can be adapted for graphs and then employed in cryptanalysis of group-based cryptographic schemes.

Finally, we note that this book is a substantial extension of our earlier introductory book [194] that was based on lecture notes for the Advanced Course on Group-Based Cryptography held at the CRM, Barcelona in May 2007.

Part 1

Background on Groups, Complexity, and Cryptography

In this part of the book we give necessary background on public-key cryptography, combinatorial group theory, and computational complexity. This background is, of course, very limited and is tailored to our needs in subsequent parts of the book.

Public-key cryptography is a relatively young area, but it has been very active since its official beginning in 1976, and by now there are great many directions of research within this area. We do not survey these directions in the present book; instead, we focus on a few of the most basic, fundamental areas within public-key cryptography, namely on key establishment, encryption, and authentication.

Combinatorial group theory, by contrast, is a rather old (over 100 years old) and established area of mathematics. Since in this book we use group theory in connection with cryptography, it is not surprising that our focus here is on algorithmic problems. Thus, in this chapter we give background on several algorithmic problems, some of them classical (known as *Dehn's problems*), others relatively new; some of them, in fact, take their origin in cryptology.

Probably nobody doubts the importance of complexity theory. This area is younger than combinatorial group theory, but older than public-key cryptography, and it is hard to name an area of mathematics or theoretical computer science that would have more applications these days than complexity theory does. In this chapter, we give background on foundations of computability and complexity: Turing machines, stratification, and complexity classes.

CHAPTER 1

Background on Public-Key Cryptography

In this chapter we describe, very briefly, some classical concepts and cryptographic primitives that were the inspiration behind new, “non-commutative”, primitives discussed in our Chapter 4. It is not our goal here to give a comprehensive survey of all or even of the most popular public key cryptographic primitives in use today, but just of those relevant to the main theme of our book, which is using non-commutative groups in cryptography. We make an exception for the RSA since it is the most common public key cryptosystem in use today; although we are unaware of any non-commutative analogs of the idea(s) behind the RSA. We also make an exception for Rabin’s cryptosystem [230] because it was the first public key cryptosystem where, unlike in the RSA, recovering the entire plaintext from the ciphertext could be actually *proved* to be as hard as factoring.

For a comprehensive survey of “commutative” cryptographic primitives, we refer the reader to numerous monographs on the subject, e.g. [90], [190], [266].

Here we discuss some basic concepts very briefly, without giving formal definitions, but emphasizing intuitive ideas instead.

First of all, there is a fundamental difference between public key (or *asymmetric*) cryptographic primitives introduced in 1976 [53] and *symmetric* ciphers that had been in use since Caesar (or even longer). In a symmetric cipher, knowledge of the decryption key is equivalent to, or often exactly equal to, knowledge of the encryption key. This implies that two communicating parties need to have an agreement on a shared secret before they engage in communication through an open channel.

By contrast, knowledge of encryption and decryption keys for asymmetric ciphers are not equivalent (by any feasible computation). For example, the decryption key might be kept secret, while the encryption key is made public, allowing many different people to encrypt, but only one person to decrypt. Needless to say, this kind of arrangement is of paramount importance for e-commerce, in particular, for electronic shopping, when no pre-existing shared secret is possible.

In the core of most public key cryptographic primitives there is an alleged practical irreversibility of some process, usually referred to as a *trapdoor function*, which is a function that is easy to compute in one direction, yet believed to be difficult to compute in the opposite direction (finding its inverse) without special information, called the “trapdoor”. For example, the RSA cryptosystem uses the fact that, while it is not hard to compute the product of two large primes, to *factor* a very large integer into its prime factors seems to be very hard. Another, perhaps even more intuitively obvious, example is that of the function $f(x) = x^2$. It is rather easy to compute in many reasonable (semi)groups, but the inverse function \sqrt{x} is much less friendly. This fact is exploited in Rabin’s cryptosystem, with the multiplicative semigroup of \mathbb{Z}_n (n composite) as the platform. A trapdoor function

may be also called a “one-way function with trapdoor”. For a rigorous definition of a one-way function we refer the reader to [268]; here we just say that there should be an efficient (which usually means polynomial-time with respect to the complexity of an input) way to compute this function, but no visible (probabilistic) polynomial-time algorithm for computing the inverse function on “most” inputs. The meaning of “most” is made more precise in Part 3 of the present book.

At the time of this writing, the prevailing tendency in public-key cryptography is to go wider rather than deeper. Applications of public-key cryptography now include digital signatures, secret sharing, multiparty secure computation, etc., etc. In this book, however, the focus is on cryptographic *primitives* and, in particular, on the ways for two parties (traditionally called Alice and Bob) to establish a common secret key without any prior arrangement. We call relevant procedures *key establishment protocols*. We note that, once a common secret key is established, Alice and Bob are in the realm of symmetric cryptography which has obvious advantages; in particular, encryption and decryption can be made very efficient once the parties have a shared secret. In the next section, we show a universal way of arranging encryption based on a common secret key. As any universal procedure, it is far from being perfect, but it is useful to keep in mind.

1.1. From key establishment to encryption

Suppose Alice and Bob share a secret key K , which is an element of a set \mathcal{S} (usually called the *key space*).

Let $H : \mathcal{S} \rightarrow \{0, 1\}^n$ be any (public) function from the set \mathcal{S} to the set of bit strings of length n . It is reasonable to have n sufficiently large, say, at least $\log_2 |\mathcal{S}|$ if \mathcal{S} is finite, or whatever your computer can afford if \mathcal{S} is infinite. Such functions are sometimes called *hash functions*. In other situations hash functions are used as compact representations, or digital fingerprints, of data and to provide message integrity.

Encryption: Bob encrypts his message $m \in \{0, 1\}^n$ as

$$E(m) = m \oplus H(K),$$

where \oplus is addition modulo 2.

Decryption: Alice computes:

$$(m \oplus H(K)) \oplus H(K) = m \oplus (H(K) \oplus H(K)) = m,$$

thus recovering the message m .

Note that this encryption has an *expansion factor* of 1, i.e., the encryption of a message is as long as the message itself. This is quite good, especially compared to the encryption in our Section 7.1, say, where the expansion factor is on the order of hundreds; this is the price one has to pay for security against a computationally superior adversary.

1.2. The Diffie-Hellman key establishment

It is rare that the beginning of a whole new area of science can be traced back to one particular paper. This is the case with public-key cryptography; it started with the seminal paper [53]. We quote from Wikipedia: “Diffie-Hellman key agreement was invented in 1976 . . . and was the first practical method for establishing a shared secret over an unprotected communications channel.” In 2002 [131], Martin

Hellman gave credit to Merkle as well: “The system . . . has since become known as Diffie-Hellman key exchange. While that system was first described in a paper by Diffie and me, it is a public-key distribution system, a concept developed by Merkle, and hence should be called ‘Diffie-Hellman-Merkle key exchange’ if names are to be associated with it. I hope this small pulpit might help in that endeavor to recognize Merkle’s equal contribution to the invention of public-key cryptography.”

U. S. Patent 4,200,770, now expired, describes the algorithm, and credits Hellman, Diffie, and Merkle as inventors.

The simplest, and original, implementation of the protocol uses the multiplicative group of integers modulo p , where p is prime and g is primitive mod p . A more general description of the protocol uses an arbitrary finite cyclic group.

- (1) Alice and Bob agree on a finite cyclic group G and a generating element g in G . We will write the group G multiplicatively.
- (2) Alice picks a random natural number a and sends g^a to Bob.
- (3) Bob picks a random natural number b and sends g^b to Alice.
- (4) Alice computes $K_A = (g^b)^a = g^{ba}$.
- (5) Bob computes $K_B = (g^a)^b = g^{ab}$.

Since $ab = ba$ (because \mathbb{Z} is commutative), both Alice and Bob are now in possession of the same group element $K = K_A = K_B$ which can serve as the shared secret key.

The protocol is considered secure against eavesdroppers if G and g are chosen properly. The eavesdropper, Eve, must solve the *Diffie-Hellman problem* (recover g^{ab} from g^a and g^b) to obtain the shared secret key. This is currently considered difficult for a “good” choice of parameters (see e.g. [190] for details).

An efficient algorithm to solve the *discrete logarithm problem* (i.e., recovering a from g and g^a) would obviously solve the Diffie-Hellman problem, making this and many other public-key cryptosystems insecure. However, it is not known whether or not the discrete logarithm problem is *equivalent* to the Diffie-Hellman problem.

We note that there is a “brute force” method for solving the discrete logarithm problem: the eavesdropper Eve can just go over natural numbers n from 1 up one at a time, compute g^n and see whether she has a match with the transmitted element. This will require $O(|g|)$ multiplications, where $|g|$ is the order of g . Since in practical implementations $|g|$ is typically about 10^{300} , this method is considered computationally infeasible.

This raises a question of computational efficiency for legitimate parties: on the surface, it looks like legitimate parties, too, have to perform $O(|g|)$ multiplications to compute g^a or g^b . However, there is a faster way to compute g^a for a particular a by using the “square-and-multiply” algorithm, based on the binary form of a . For example, $g^{22} = (((g^2)^2)^2 \cdot (g^2)^2 \cdot g^2$. Thus, to compute g^a , one actually needs $O(\log_2 a)$ multiplications, which is quite feasible given the magnitude of a .

1.3. The ElGamal cryptosystem

The ElGamal cryptosystem [68] is a public-key cryptosystem which is based on the Diffie-Hellman key establishment (see the previous section). The ElGamal protocol is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm is a variant of the ElGamal signature scheme, which should not be confused with the ElGamal encryption protocol that we describe below.

- (1) Alice and Bob agree on a finite cyclic group G and a generating element g in G .
- (2) Alice (the receiver) picks a random natural number a and publishes $c = g^a$.
- (3) Bob (the sender), who wants to send a message $m \in G$ (called a “plaintext” in cryptographic lingo) to Alice, picks a random natural number b and sends two elements, $m \cdot c^b$ and g^b , to Alice. Note that $c^b = g^{ab}$.
- (4) Alice recovers $m = (m \cdot c^b) \cdot ((g^b)^a)^{-1}$.

A notable feature of the ElGamal encryption is that it is *probabilistic*, meaning that a single plaintext can be encrypted to many possible ciphertexts.

We also point out that the ElGamal encryption has an average expansion factor of 2. (Compare this to the encryption described in Section 1.1.)

1.4. The RSA cryptosystem

RSA is an algorithm for public-key encryption that is widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations. We include its description in this book as a tribute to its popularity and influence, but we also have to admit that ideas behind the RSA algorithm do not have any non-commutative ramifications known at the time of this writing. This is probably due to the fact that RSA’s mechanism is based on Euler’s generalization of Fermat’s little theorem, an elementary fact from number theory that does not yet seem to have any non-commutative analogs.

Below we give a description of the RSA encryption algorithm. Here Bob is the encrypter and Alice the decrypter (although, surprisingly, you will not find the latter word in the dictionary).

- (1) Alice’s private key is a pair of large primes p, q , and her public key consists of: (1) the product $n = pq$; (2) an integer e such that $1 < e < \varphi(n)$, and e and $\varphi(n)$ are relatively prime. Here $\varphi(n) = (p - 1)(q - 1)$, the Euler function of n .
- (2) If Bob wants to encrypt his message m , which is an integer, $0 < m < n$, he computes $c \equiv m^e \pmod{n}$ and sends c to Alice.
- (3) To decrypt, Alice first finds an integer d such that $de \equiv 1 \pmod{\varphi(n)}$. Then she computes:

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.$$

Now, since $ed = 1 + k\varphi(n)$, one has

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m(m^k)^{\varphi(n)} \equiv m \pmod{n}.$$

The last congruence follows directly from Euler’s generalization of Fermat’s little theorem if m is relatively prime to n . By using the Chinese remainder theorem it can be shown that the equations hold for all m .

A lot of research has been done on how to select “good” private keys p and q (i.e., those that make the RSA protocol secure or, more accurately, not visibly insecure), but this subject is outside of the scope of our book. As a sample of what is known by now, we can mention that if either $p - 1$ or $q - 1$ has only small prime factors, then n can be factored quickly by Pollard’s “ $p - 1$ algorithm”, and these values of p or q should therefore be discarded.

1.5. Rabin's cryptosystem

Rabin's cryptosystem [230] was the first public-key cryptosystem where recovering the entire plaintext from the ciphertext could be proved to be as hard as factoring.

- (1) Alice's private key is a pair of large primes p, q , where $p \equiv q \equiv 3 \pmod{4}$, and her public key is the product $n = pq$.
- (2) If Bob wants to encrypt his message m , which is an integer, $0 < m < n$, he computes $c \equiv m^2 \pmod{n}$ and sends c to Alice.
- (3) Alice first computes:

$$m_p = c^{\frac{(p+1)}{4}} \pmod{p}$$

and

$$m_q = c^{\frac{(q+1)}{4}} \pmod{q}.$$

Then she computes the following four square roots of $c \pmod{n}$:

$$\pm r = (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \pmod{n}$$

$$\pm s = (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \pmod{n}.$$

Here y_p and y_q , such that $y_p \cdot p + y_q \cdot q = 1$, can be found by using Euclidean algorithm. It can be verified directly that $\pm r$ as well as $\pm s$ are square roots of c modulo p and modulo q , and therefore also modulo n , by the Chinese remainder theorem.

The condition $p \equiv q \equiv 3 \pmod{4}$ makes it easier to compute square roots, but it is not an absolute requirement. Rabin [230] proposes to find the square roots modulo primes by using a special case of Berlekamp's algorithm.

We note that breaking this system completely, i.e., obtaining the private keys p, q is apparently equivalent for the adversary to have the ability to efficiently compute square roots modulo $n = pq$. The latter is known to be equivalent to the existence of a probabilistic algorithm for factoring n with polynomial expected running time. Indeed, if $n = pq$, then, given a square $x^2 \pmod{n}$, there are four different square roots, call them $\pm x$ and $\pm y$. If we know x and y , then

$$(x - y)(x + y) = x^2 - y^2 = 0 \pmod{n}.$$

Therefore, $n = pq$ divides $(x - y)(x + y)$, so either p divides $(x + y)$ and q divides $(x - y)$ or vice versa. In either case we can easily find one of the prime factors of n by computing $\text{g.c.d.}(x + y, n)$ using Euclidean algorithm.

1.6. Authentication

Authentication is a process of verifying the digital identity of the sender of a communication. Of particular interest in public-key cryptography are *zero-knowledge* proofs of identity. This means that if the identity is true, no malicious verifier learns anything other than this fact. Thus, one party (the prover) wants to prove its identity to a second party (the verifier) via some secret information (a private key), but does not want anybody to learn anything about this secret.

Many key establishment protocols can be (slightly) modified to become public-key authentication protocols. We illustrate this on the example of the Diffie-Hellman key establishment protocol (see Section 1.2).

Suppose Alice is the prover and Bob is the verifier, so that Alice wants to convince Bob that she knows a secret without revealing the secret itself. Alice's

public key is a finite cyclic group G , a generating element g in G , and g^a for some random natural number a . This number a is Alice's private key. The authentication protocol itself is the following sequence of steps.

- (1) Bob picks a random natural number b and sends a *challenge* g^b to Alice.
- (2) Alice responds with a proof $P = (g^b)^a = g^{ba}$.
- (3) Bob verifies: $(g^a)^b = P?$.

We see that this protocol is almost identical to the Diffie-Hellman key establishment protocol. Later, in Chapter 8, we will see examples of “independent” authentication protocols, which are not just modifications of key establishment protocols. Right now we are going to describe one classical authentication protocol of that sort from “commutative cryptography”. It comes from [74].

1.6.1. The Feige-Fiat-Shamir scheme. Just like the RSA (see our Section 1.4), this protocol relies in its security on the difficulty of finding a factorization of an integer in a product of two large primes.

Alice's (long-term) private key consists of: (1) a pair of large primes p, q ; (2) integers s_1, \dots, s_k with $\gcd(s_i, pq) = 1$. Her public key consists of: (1) the product $n = pq$; (2) integers $v_i \equiv s_i^2 \pmod{n}$. The authentication protocol itself is the following sequence of steps. (Note the *commitment* step – this is a very useful idea!)

- (1) *Commitment.* Alice chooses a random integer r , computes $x \equiv r^2 \pmod{n}$ and sends x to Bob.
- (2) *Challenge.* Bob chooses a_1, \dots, a_k , where a_i equals 0 or 1, and sends these numbers to Alice.
- (3) *Response.* Alice computes $y \equiv rs_1^{a_1}s_2^{a_2} \cdots s_k^{a_k} \pmod{n}$ and sends y to Bob.
- (4) *Verification.* Bob checks that $y^2 \equiv \pm xv_1^{a_1}v_2^{a_2} \cdots v_k^{a_k} \pmod{n}$. If the congruence holds, he accepts authentication. If not, then rejects.

This protocol is repeated several times, each time with fresh randomness, to prevent the adversary from guessing the values of a_i and thus impersonating Alice (beginning with the commitment step).

We note that the most straightforward way for the adversary to break this scheme completely, i.e., to obtain the (long-term) private key would be to (efficiently) compute square roots modulo $n = pq$. The latter is known to be equivalent to the existence of a probabilistic algorithm for factoring n with polynomial expected running time; see our Section 1.5.

CHAPTER 2

Background on Combinatorial Group Theory

In this chapter, we first give the definition of a free group, followed by the definition of a presentation of a group by generators and relators, and then, in Section 2.3, describe algorithmic problems of group theory that will be exploited in Chapters 4 and 15 of this book. Having in mind applications to cryptology, we emphasize *search problems*, as compared to *decision problems* traditionally studied in group theory.

in Sections 2.4 and 2.5, we give a brief exposition of several classical techniques in combinatorial group theory, namely methods of Nielsen, Schreier, and Tietze. We do not go into details here because there are two very well established monographs where a complete exposition of these techniques is given. For an exposition of Nielsen's and Schreier's methods, we recommend [179], whereas [177] has, in our opinion, a better exposition of Tietze's methods.

In the concluding Section 2.6, we touch upon *normal forms* of group elements as a principal hiding mechanism for cryptographic protocols.

2.1. Basic definitions and notation

Let G be a group. If H is a subgroup of G , we write $H \leq G$; if H is a normal subgroup of G , we write $H \trianglelefteq G$. For a subset $A \subseteq G$, by $\langle A \rangle$ we denote the subgroup of G generated by A (the intersection of all subgroups of G containing A). It is easy to see that

$$\langle A \rangle = \{a_{i_1}^{\varepsilon_1}, \dots, a_{i_n}^{\varepsilon_n} \mid a_{i_j} \in A, \varepsilon_j \in \{1, -1\}, n \in \mathbb{N}\}.$$

Let X be an arbitrary set. A *word* w in X is a finite (possibly empty) sequence of elements that we write as $w = y_1 \dots y_n$, $y_i \in X$. The number n is called the *length* of the word w ; we denote it by $|w|$. We denote the empty word by ε and put $|\varepsilon| = 0$. Then, let $X^{-1} = \{x^{-1} \mid x \in X\}$, where x^{-1} is just a formal expression obtained from x and -1 . If $x \in X$, then the symbols x and x^{-1} are called *literals* in X . Denote by $X^{\pm 1} = X \cup X^{-1}$ the set of all literals in X .

An expression of the form

$$(1) \quad w = x_{i_1}^{\varepsilon_1} \cdots x_{i_n}^{\varepsilon_n},$$

where $x_{i_j} \in X, \varepsilon_j \in \{1, -1\}$ is called a *group word* in X . So a group word in X is just a word in the alphabet $X^{\pm 1}$.

A group word $w = y_1 \cdots y_n$ is *reduced* if for any $i = i, \dots, n-1$, $y_i \neq y_{i+1}^{-1}$, that is, w does not contain a subword of the form yy^{-1} for any literal $y \in X^{\pm 1}$. We assume that the empty word is reduced.

If $X \subseteq G$, then every group word $w = x_{i_1}^{\varepsilon_1} \cdots x_{i_n}^{\varepsilon_n}$ in X determines a unique element of G which is equal to the product $x_{i_1}^{\varepsilon_1} \cdots x_{i_n}^{\varepsilon_n}$ of the elements $x_{i_j}^{\varepsilon_j} \in G$. By convention, the empty word ε determines the identity 1 of G .

DEFINITION 2.1.1. A group G is called a free group if there is a generating set X of G such that every non-empty reduced group word in X defines a non-trivial element of G .

In this case X is called a *free basis* of G and G is called a *free group on X* , or a *group freely generated by X* . It follows from the definition that every element of $F(X)$ can be defined by a reduced group word in X . Moreover, different reduced words in X define different elements of G .

Free groups have the following *universal property*:

THEOREM 2.1.2. *Let G be a group with a generating set $X \subseteq G$. Then G is free on X if and only if the following universal property holds: every map $\varphi : X \rightarrow H$ from X into a group H can be extended to a unique homomorphism*

$$\varphi^* : G \rightarrow H,$$

so that the diagram below is commutative:

$$\begin{array}{ccc} X & \xrightarrow{i} & G \\ & \searrow \varphi & \downarrow \varphi^* \\ & & H \end{array}$$

(here $X \xrightarrow{i} G$ is the natural inclusion of X into G).

COROLLARY 2.1.3. *Let G be a free group on X . Then the identity map $X \rightarrow X$ extends to an isomorphism $G \rightarrow F(X)$.*

This corollary allows us to identify a free group freely generated by X with the group $F(X)$ of reduced group words in X . In what follows we usually call the group $F(X)$ a free group on X .

2.2. Presentations of groups by generators and relators

The universal property of free groups allows one to describe arbitrary groups in terms of *generators* and *relators*.

Let G be a group with a generating set X . By the universal property of free groups there exists a homomorphism $\psi : F(X) \rightarrow G$ such that $\psi(x) = x$ for $x \in X$. It follows that ψ is onto, so by the first isomorphism theorem

$$G \simeq F(X)/\ker(\psi).$$

In this case $\ker(\psi)$ is viewed as the set of relators of G , and a group word $w \in \ker(\psi)$ is called a *relator* of G in generators X . If a subset $R \subseteq \ker(\psi)$ generates $\ker(\psi)$ as a normal subgroup of $F(X)$ then it is termed a set of *defining relators* of G relative to X . The pair $\langle X \mid R \rangle$ is called a *presentation* of a group G ; it determines G uniquely up to isomorphism. The presentation $\langle X \mid R \rangle$ is finite if both sets X and R are finite. A group is *finitely presented* if it has at least one finite presentation. Presentations provide a universal method to describe groups. In particular, finitely presented groups admit finite descriptions, e.g.,

$$G = \langle x_1, x_2, \dots, x_n \mid r_1, r_2, \dots, r_k \rangle.$$

All finitely generated abelian groups are finitely presented (a group G is *abelian*, or commutative, if $ab = ba$ for all $a, b \in G$). Other examples of finitely presented groups include finitely generated nilpotent groups, braid groups (Section 5.1), Thompson’s group (Section 5.2).

2.3. Algorithmic problems of group theory: decision, witness, search

Algorithmic problems of (semi)group theory that we consider in this section are of three different kinds:

- (1) *Decision problems* are problems of the following nature: given a property \mathcal{P} and an object \mathcal{O} , find out whether or not the object \mathcal{O} has the property \mathcal{P} .
- (2) *Witness problems* are of the following nature: given a property \mathcal{P} and an object \mathcal{O} with the property \mathcal{P} , find a proof (a “witness”) of the fact that \mathcal{O} has the property \mathcal{P} . Such a proof does not necessarily have to produce anything “material”; for example, we mentioned in the Introduction that one of the ways to prove invertibility of a matrix over a field is reducing it by elementary row or column operations to the identity matrix. This way does not by itself produce the inverse of a given matrix, although, of course, using little extra effort it will.
- (3) *Search problems* are a special case of witness problems, and some of them are important for applications to cryptography: given a property \mathcal{P} and the information that there are objects with the property \mathcal{P} , find a proof *of a particular kind* establishing the property \mathcal{P} . Usually, the most “natural” proof is sought in this context; for example, given two conjugate elements of a group, find a conjugator.

All decision problems in group theory have a “companion” witness version, and most of them also have a natural search version. We are now going to discuss several particular algorithmic problems of group theory that have been used in cryptography.

2.3.1. The word problem.

The *word (decision) problem* (WP) is: given a recursive presentation of a group G and an element $g \in G$, find out whether or not $g = 1$ in G .

From the very description of the word problem we see that it consists of two parts: “whether” and “not”. We call them the “yes” and “no” parts of the word problem, respectively. If a group is given by a recursive presentation in terms of generators and relators, then the “yes” part of the word problem has a recursive solution:

PROPOSITION 2.3.1. *Let $\langle X; R \rangle$ be a recursive presentation of a group G . Then the set of all words $g \in G$ such that $g = 1$ in G is recursively enumerable.*

The *word witness problem* is: given that a word w is in the normal closure of R , find a proof (a “witness”) of that fact.

The *word search problem* (WSP) is: given a recursive presentation of a group G and an element $g = 1$ in G , find a presentation of g as a product of conjugates of defining relators and their inverses.

We note that the word search problem always has a recursive solution because one can recursively enumerate all products of defining relators, their inverses and conjugates. However, the number of factors in such a product required to represent a word of length n which is equal to 1 in G , can be very large compared to n ; in particular, there are groups G with efficiently solvable word problem and words w of length n equal to 1 in G , such that the number of factors in any factorization of w into a product of defining relators, their inverses and conjugates is not bounded by any tower of exponents in n ; see [223]. Furthermore, if in a group G the word problem is recursively unsolvable, then the length of a proof verifying that $w = 1$ in G is not bounded by any recursive function of the length of w .

2.3.2. The conjugacy problem.

The next problems of interest to us are:

The *conjugacy (decision) problem* (CP) is: given a recursive presentation of a group G and two elements $g, h \in G$, find out whether or not there is an element $x \in G$ such that $x^{-1}gx = h$.

Again, just as the word problem, the conjugacy problem consists of the “yes” and “no” parts, with the “yes” part always recursive because one can recursively enumerate all conjugates of a given element.

The *conjugacy witness problem* is: given two words w_1, w_2 representing conjugate elements of G , find a proof (a “witness”) of the fact that the elements are conjugate.

The *conjugacy search problem* (CSP) is: given a recursive presentation of a group G and two conjugate elements $g, h \in G$, find a particular element $x \in G$ such that $x^{-1}gx = h$.

As we have already mentioned, the conjugacy search problem always has a recursive solution because one can recursively enumerate all conjugates of a given element, but as with the word search problem, this kind of solution can be extremely inefficient.

2.3.3. The decomposition and factorization problems.

One of the natural ramifications of the conjugacy search problem is the following:

The *decomposition search problem*: given a recursive presentation of a group G , two recursively generated subgroups $A, B \leq G$, and two elements $g, h \in G$, find two elements $x \in A$ and $y \in B$ that would satisfy $x \cdot g \cdot y = h$, provided at least one such pair of elements exists.

We note that *some* x and y satisfying the equality $x \cdot g \cdot y = h$ always exist (e.g., $x = 1$, $y = g^{-1}h$), so the point is to have them satisfy the conditions $x \in A$, $y \in B$. We therefore will not usually refer to this problem as a *subgroup-restricted* decomposition search problem because it is always going to be subgroup-restricted; otherwise it does not make much sense.

A special case of the decomposition search problem, where $A = B$, is also known as the *double coset problem*.

Clearly, the decomposition problem also has the decision version. So far, it has not been used in cryptography.

One more special case of the decomposition search problem, where $g = 1$, deserves special attention.

The *factorization problem*: given an element w of a recursively presented group G and two subgroups $A, B \leq G$, find out whether or not there are two elements $a \in A$ and $b \in B$ such that $a \cdot b = w$.

The *factorization search problem*: given an element w of a recursively presented group G and two recursively generated subgroups $A, B \leq G$, find any two elements $a \in A$ and $b \in B$ that would satisfy $a \cdot b = w$, provided at least one such pair of elements exists.

There are relations between the algorithmic problems discussed so far, which we summarize in the following.

PROPOSITION 2.3.2. *Let G be a recursively presented group.*

- (1) *If the conjugacy problem in G is solvable, then the word problem is solvable, too.*
- (2) *If the conjugacy search problem in G is solvable, then the decomposition search problem is solvable for commuting subgroups $A, B \leq G$ (i.e., $ab = ba$ for all $a \in A, b \in B$).*
- (3) *If the conjugacy search problem in G is solvable, then the factorization search problem is solvable for commuting subgroups $A, B \leq G$.*

The first statement of this proposition is obvious since conjugacy to the identity element 1 is the same as equality to 1. Two other statements are not immediately obvious; proofs are given in our Section 4.6.

2.3.4. The membership problem. Now we are coming to the next trio of problems.

The subgroup *membership (decision) problem*: given a recursively presented group G , a subgroup $H \leq G$ generated by h_1, \dots, h_k , and an element $g \in G$, find out whether or not $g \in H$.

Again, the membership problem consists of the “yes” and “no” parts, with the “yes” part always recursive because one can recursively enumerate all elements of a subgroup given by finitely many generators.

We note that the membership problem also has a less descriptive name, “the generalized word problem”.

The subgroup *membership witness problem* is: given a group G , a subgroup H generated by h_1, \dots, h_k , and an element $h \in H$, find a proof of the fact that $h \in H$.

The *membership search problem*: given a recursively presented group G , a subgroup $H \leq G$ generated by h_1, \dots, h_k , and an element $h \in H$, find an expression of h in terms of h_1, \dots, h_k .

In Section 2.4, we are going to show how the membership problem can be solved for any finitely generated subgroup of any free group.

2.3.5. The isomorphism problem. Finally, we mention the isomorphism problem that will be important in our Chapter 15.

The *isomorphism (decision) problem* is: given two finitely presented groups G_1 and G_2 , find out whether or not they are isomorphic.

We note that Tietze's method described in our Section 2.5 provides a recursive enumeration of all finitely presented groups isomorphic to a given finitely presented group, which implies that the “yes” part of the isomorphism problem is always recursive.

On the other hand, specific instances of the isomorphism problem *may* provide examples of group-theoretic decision problems both the “yes” and “no” parts of which are non-recursive. In Section 2.3.6 below we offer a candidate problem of that kind.

2.3.6. More on search/witness problems. Now we make one general observation. Decision problems usually naturally split into the “yes” and “no” parts, and the “yes” part of most popular decision problems in group theory usually has a recursive solution; for example, the “yes” part of the word problem has a recursive solution because, given a recursive presentation of a group G , the set of all words w such that $w = 1$ in G is recursively enumerable. The same can be said about the “yes” part of the conjugacy problem, the isomorphism problem, etc. At the same time, the “no” part of these problems is typically *not* recursively enumerable in general. However, one can still ask for a proof (a “witness”) of the fact that, say, a given word w is not equal to 1 in G , or a given pair of words represent non-conjugate elements of G , etc. We call the corresponding search problems the non-identity witness problem (because calling it the “non-word witness problem” would be kind of ridiculous) and the non-conjugacy witness problem, respectively. Similarly, one can consider non-membership witness problem, non-isomorphism witness problem, etc.

As we have pointed out before, in general there is no recursive procedure for enumerating all words w such that $w \neq 1$ in G , or all words representing elements that do not belong to a given subgroup of G , etc. Of course, if, for example, G has solvable word problem, then enumerating all words $w \neq 1$ in G is possible by an obvious procedure. However, what we are looking for here is a more general way of proving $w \neq 1$ that would be applicable also to “many” groups with unsolvable word problem. One fairly general approach to proving $w \neq 1$ in G would be to exhibit a “smooth” factor group of G (often just the abelianization $G/[G, G]$ would work) where $w \neq 1$. This approach is discussed in detail in [146], and it also works for the non-conjugacy witness problem and for the non-membership witness problem. Still, it would be quite interesting to find other sufficiently general methods for proving non-identity, non-conjugacy, etc.

At the same time, it would be quite interesting (and useful) to have a general way (applicable to *any* non-trivial group G) of proving $w \neq 1$ at least for some particular words w (depending on G). This may be regarded as a special case of the non-isomorphism witness problem, namely, as proving that a given group is non-trivial:

PROBLEM 2.3.3. (M. Chiodo) Is there a general procedure to produce a non-trivial element from a finite presentation of a non-trivial group?

This problem is discussed in [39], where a special case is settled; namely, it is shown that there is no general procedure to pick a non-trivial generator from a finite presentation of a non-trivial group. We emphasize here the importance for cryptographic applications of any progress on Problem 2.3.3 in the positive direction.

Building on the same idea, one can also ask:

PROBLEM 2.3.4. Is there a general procedure to produce an element that does not belong to a given (finitely generated) proper subgroup of a given finitely presented group?

To conclude this section, we point out that some specific problems may provide examples of natural group-theoretic decision problems with both the “yes” and “no” parts non-recursive, which would be of great interest. Here we can offer some candidate problems of that kind.

PROBLEM 2.3.5. Is the set of all finitely presented metabelian groups recursively enumerable?

A group is called metabelian if its commutator subgroup is abelian. Metabelian groups and their properties are discussed in our Section 5.5. The set of finitely presented non-metabelian groups is known to be non-recursive; see e.g. [1]. At the same time, there is no obvious way to recursively enumerate all finitely presented metabelian groups because many metabelian groups are not finitely presented, so it is not clear how to specifically enumerate just the finitely presented ones.

This problem may provide an example of a natural search problem in group theory that is algorithmically unsolvable. There are many algorithmically unsolvable decision problems in group theory (see e.g. [1] or [158]), but the following might be the first natural example of an unsolvable witness problem:

PROBLEM 2.3.6. Given a finitely presented group and the information that it is metabelian, find a proof (a “witness”) of that fact.

Another interesting decision problem that might have both the “yes” and “no” parts non-recursive is: Given two finitely presented groups G_1 and G_2 , is there an injective homomorphism (an embedding) of G_1 into G_2 ? This problem is known to have a negative answer, but the point is, again, that it might have both the “yes” and “no” parts non-recursive, as was suggested to us by D. Groves. We note that without the word “injective”, the “yes” part of this problem would have an affirmative answer, i.e., all homomorphisms of G_1 into G_2 are recursively enumerable.

Thus, we have the following witness problem that may be algorithmically unsolvable:

PROBLEM 2.3.7 (D. Groves). Given two finitely presented groups G_1 and G_2 and the information that there is an injective homomorphism (an embedding) of G_1 into G_2 , find a proof (a “witness”) of that fact.

We note that Chiodo [39] has recently proved that there is no algorithm that, on input of finite presentations of two groups and information that one of them embeds into the other, outputs an explicit embedding. This does not immediately provide a negative answer to Problem 2.3.7 above because there might be other ways to prove the existence of an embedding (for example, if G_1 is a cyclic group of order n , then finding an element of order n in G_2 would be such a proof), but this is a serious argument in favor of a negative answer nonetheless.

Another example of a similar kind was reported in the same paper [39]: given a finite presentation of a group G and information that G has an element of a finite order $n \geq 2$, there is, in general, no algorithm to find a particular element of order n . In fact, it was shown in [39] that there is no algorithm to even find *any* torsion element in G . Again, this does not necessarily imply that there is no *proof* (or “witness”) of the existence of an element of order n .

2.4. Nielsen's and Schreier's methods

Let $F = F_n$ be the free group of a finite rank $n \geq 2$ with a set $X = \{x_1, \dots, x_n\}$ of free generators. Let $Y = \{y_1, \dots, y_m\}$ be an arbitrary finite set of elements of the group F . Consider the following elementary transformations that can be applied to Y :

- (N1): y_i is replaced by $y_i y_j$ or by $y_j y_i$ for some $j \neq i$;
- (N2): y_i is replaced by y_i^{-1} ;
- (N3): y_i is replaced by some y_j , and at the same time y_j is replaced by y_i ;
- (N4): delete some y_i if $y_i = 1$.

It is understood that y_j does not change if $j \neq i$.

Every finite set of reduced words of F_n can be carried by a finite sequence of Nielsen transformations to a *Nielsen-reduced* set $U = \{u_1, \dots, u_k\}$; i.e., a set such that for any triple v_1, v_2, v_3 of the form $u_i^{\pm 1}$, the following three conditions hold:

- (i) $v_1 \neq 1$;
- (ii) $v_1 v_2 \neq 1$ implies $|v_1 v_2| \geq |v_1|, |v_2|$;
- (iii) $v_1 v_2 \neq 1$ and $v_2 v_3 \neq 1$ implies $|v_1 v_2 v_3| > |v_1| - |v_2| + |v_3|$.

It is easy to see that if $U = (u_1, u_2, \dots, u_k)$ is Nielsen-reduced, then the subgroup of F_n generated by U is free with a basis U ; see e.g. [179].

One might notice that some of the transformations (N1)–(N4) are redundant; i.e., they are compositions of other ones. The reason behind that will be explained below.

We say that two sets Y and \tilde{Y} are Nielsen equivalent if one of them can be obtained from another by applying a sequence of transformations (N1)–(N4). It was proved by Nielsen that two sets Y and \tilde{Y} generate the same subgroup of the group F if and only if they are Nielsen equivalent. This result is now one of the central points in combinatorial group theory.

Note, however, that this result alone does not give an *algorithm* for deciding whether or not Y and \tilde{Y} generate the same subgroup of F . To obtain an algorithm, we need to somehow define the *complexity* of a given set of elements and then show that a sequence of Nielsen transformations (N1)–(N4) can be arranged so that this complexity decreases (or, at least, does not increase) *at every step* (this is where we may need “redundant” elementary transformations!).

This was also done by Nielsen; the complexity of a given set $Y = \{y_1, \dots, y_m\}$ is just the sum of the lengths of the words y_1, \dots, y_m . Now the algorithm is as follows. Reduce both sets Y and \tilde{Y} to Nielsen-reduced sets. This procedure is finite because the sum of the lengths of the words decreases at every step. Then solve the membership problem for every element of Y in the subgroup generated by \tilde{Y} and vice versa.

Nielsen's method therefore yields (in particular) an algorithm for deciding whether or not a given endomorphism of a free group of finite rank is an automorphism.

Another classical method that we sketch in this section is that of Schreier. We are going to give a brief exposition of this method here in a special case of a free group; however, it is valid for arbitrary groups as well.

Let H be a subgroup of F . A *right coset representative function* for F (on the generators x_i) modulo H is a mapping of words in x_i , $w(x_1, x_2, \dots) \rightarrow \overline{w}(x_1, x_2, \dots)$, where the $\overline{w}(x_1, x_2, \dots)$ form a right coset representative system

for F modulo H , which contains the empty word, and where $\overline{w}(x_1, x_2, \dots)$ is the representative of the coset of $w(x_1, x_2, \dots)$. Then we have:

THEOREM 2.4.1. *If $w \rightarrow \overline{w}$ is a right coset function for F modulo H , then H is generated by the words*

$$ux_i \cdot \overline{ux_i}^{-1},$$

where u is an arbitrary representative and x_i is a generator of F .

This already implies, for instance, that if F is finitely generated and H is a subgroup of finite index, then H is finitely generated.

Furthermore, a *Schreier right coset function* is one for which any initial segment of a representative is again a representative. The system of representatives is then called a *Schreier system*. It can be shown that there is always some Schreier system of representatives for F modulo H . Also, there is a *minimal Schreier system*; i.e., a Schreier system in which each representative has a length not exceeding the length of any word it represents. Not every Schreier system is minimal.

EXAMPLE 2.4.2. Let F be the free group on a and b , and let H be the normal subgroup of F generated by a^2 , b^2 , and $aba^{-1}b^{-1}$. Then F/H has four cosets. The representative system $\{1, a, b, ab\}$ is Schreier, as is the representative system $\{1, a, b, ab^{-1}\}$. The representative system $\{1, a, b, a^{-1}b^{-1}\}$ is not Schreier; for, the initial segment a^{-1} of $a^{-1}b^{-1}$ is not a representative.

Using a *Reidemeister–Schreier rewriting process*, one can obtain a presentation (by generators and relators) for H . This implies, among other things, the following important result:

THEOREM 2.4.3 (Nielsen–Schreier). *Every nontrivial subgroup H of a free group F is free.*

One can effectively obtain a set of free generating elements for H ; namely, those $ux_i \cdot \overline{ux_i}^{-1}$ such that ux_i is not freely equal to a representative. Schreier obtained this set of free generators for H in 1927. In 1921, Nielsen, using quite a different method, had constructed a set of free generators for H if F was finitely generated. The generators of Nielsen are precisely those Schreier generators obtained when a minimal Schreier system is used.

We refer to [179, Chapter 3] for more details.

2.5. Tietze's method

Attempting to solve one of the major problems of combinatorial group theory, the isomorphism problem, Tietze introduced isomorphism-preserving elementary transformations that can be applied to groups presented by generators and relators.

Let

$$G = \langle x_1, x_2, \dots \mid r_1, r_2, \dots \rangle$$

be a presentation of a group $G = F/R$, where F is the ambient free group generated by x_1, x_2, \dots , and R is the normal closure of r_1, r_2, \dots ; i.e., the smallest normal subgroup of F containing r_1, r_2, \dots

The elementary transformations are of the following types.

- (T1): *Introducing a new generator:* Replace $\langle x_1, x_2, \dots \mid r_1, r_2, \dots \rangle$ by $\langle y, x_1, x_2, \dots \mid ys^{-1}, r_1, r_2, \dots \rangle$, where $s = s(x_1, x_2, \dots)$ is an arbitrary element in the generators x_1, x_2, \dots

(T2): Canceling a generator (this is the converse of (T1)): If we have a presentation of the form $\langle y, x_1, x_2, \dots | q, r_1, r_2, \dots \rangle$, where q is of the form ys^{-1} , and s, r_1, r_2, \dots are in the group generated by x_1, x_2, \dots , replace this presentation by $\langle x_1, x_2, \dots | r_1, r_2, \dots \rangle$.

(T3): Applying an automorphism: Apply an automorphism of the free group generated by x_1, x_2, \dots to all the relators r_1, r_2, \dots

(T4): Changing defining relators: Replace the set r_1, r_2, \dots of defining relators by another set r'_1, r'_2, \dots with the same normal closure. That means, each of r'_1, r'_2, \dots should belong to the normal subgroup generated by r_1, r_2, \dots , and vice versa.

Then we have the following useful result due to Tietze (see, e.g., [177]):

THEOREM 2.5.1. *Two groups $\langle x_1, x_2, \dots | r_1, r_2, \dots \rangle$ and $\langle x_1, x_2, \dots | s_1, s_2, \dots \rangle$ are isomorphic if and only if one can get from one of the presentations to the other by a sequence of transformations (T1)–(T4).*

For most practical purposes, groups under consideration are finitely presented, in which case there exists a finite sequence of transformations (T1)–(T4) taking one of the presentations to the other. Still, Theorem 2.5.1 does not give any constructive procedure for deciding in a finite number of steps whether one finite presentation can be obtained from another by Tietze transformations because, for example, there is no indication of how to select the element S in a transformation (T1). Thus Theorem 2.5.1 does not yield a solution to the isomorphism problem.

However, it has been used in many situations to derive various invariants of isomorphic groups, most notably Alexander polynomials that turned out to be quite useful in knot theory.

Also, Tietze's method gives an easy, practical way of constructing “exotic” examples of isomorphic groups that helped to refute several conjectures in combinatorial group theory. For a similar reason, Tietze transformations can be useful for “diffusing” presentations of groups in some cryptographic protocols.

2.6. Normal forms

Normal forms of group elements are principal hiding mechanisms for cryptographic protocols.

A normal form is required to have two essential properties: (1) every object under consideration must have exactly one normal form, and (2) two objects that have the same normal form must be the same up to some equivalence. The uniqueness requirement in (1) is sometimes relaxed, allowing the normal form to be unique up to some simple equivalence.

Normal forms may be “natural” and simple, but they may also be quite elaborate. We give some examples of normal forms below.

EXAMPLE 2.6.1. In the (additive) group of integers, we have many “natural” normal forms: decimal, binary, etc. These are quite good for hiding factors in a product; for example, in the product $3 \cdot 7 = 21$ we do not see 3 or 7. We make one more observation here which is important from the point of view of cryptography: if there are several different normal forms for elements of a given group, then one normal form might reveal what another one is trying to conceal. For example, the number 31 in the decimal form “looks random”, but the same number in the binary

form has a clear pattern: 11111. We re-iterate this point in Sections 5.1 and 5.2 of this book, in more complex situations.

There are also other normal forms for integers; for example, every integer is a product of primes. This factorization is not unique, but if we require, in addition, that a product of primes should be *in increasing order*, then it becomes a unique normal form.

EXAMPLE 2.6.2. In a group of matrices over a ring R , every matrix is the normal form for itself. This normal form is unique up to the equality of the entries in the ring R .

EXAMPLE 2.6.3. In some groups given by generators and defining relators (cf. our Section 2.2), there are *rewriting systems*, i.e., procedures which take a word in a given alphabet as input and transform it to another word in the same alphabet by using defining relators. This procedure terminates with the normal form of the group element represented by the input word. We give an example of rewriting system in Thompson's group in Section 5.2 of this book.

In other groups given by generators and defining relators normal forms may be based on some special (topological, or geometric, or other) properties of a given group, and not just on a rewriting system. A good example of this sort is provided by braid groups; see our Section 5.1. There are several different normal forms for elements of a braid group; the classical one, called the Garside normal form, is not even a word in generators of the group. We cannot give more details here without introducing a large amount of background material, so we just refer the interested reader to the monographs [21] and [56].

CHAPTER 3

Background on Computational Complexity

3.1. Algorithms

In all instances, if not said otherwise, we use Turing machines as our principal model of computation. In this section we briefly recall some basic definitions and notation concerning Turing machines that are used throughout this chapter.

3.1.1. Deterministic Turing machines. In this section we give basic definitions of deterministic and non-deterministic Turing machines to be used in the sequel.

DEFINITION 3.1.1. A one-tape *Turing machine* (TM) M is a 5-tuple $\langle Q, \Sigma, s, f, \delta \rangle$ where:

- Q is a finite set of *states*;
- Σ is a finite set of the *tape alphabet*;
- $s \in Q$ is the *initial state*;
- $f \in Q$ is the *final state*;
- $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}$ called the *transition function*.

Additionally, M uses a blank symbol \sqcup different from the symbols in Σ to mark the parts of the infinite tape that are not in use.

We can define the operation of a TM formally using the notion of a configuration which contains a complete description of the current state of computation. A *configuration* of M is a triple (q, w, u) , where w, u are Σ -strings and $q \in Q$.

- w is a string to the left of the head;
- u is the string to the right of the head, including the symbol scanned by the head;
- q is the current state.

We say that a configuration (q, w, u) *yields* a configuration (q', w', u') in one step, denoted by $(q, w, u) \xrightarrow{M} (q', w', u')$, if a step of the machine from the configuration (q, w, u) results in the configuration (q', w', u') . Using the relation “yields in one step” we can define relations “yields in k steps”, denoted by $\xrightarrow{M^k}$, and “yields”, denoted by $\xrightarrow{M^*}$. A sequence of configurations that M yields on the input x is called an *execution flow* of M on x .

We say that M *halts* on $x \in \Sigma^*$ if the configuration (s, ε, x) yields a configuration (f, w, u) for some Σ -strings w and u ; in this case, (f, w, u) is called a halting configuration. The number of steps M performs on a Σ -string x before it stops is denoted by $T_M(x)$. The *halting problem* for M is an algorithmic problem to determine whether M halts or not, i.e., whether $T_M(x) = \infty$ or not.

We say that a TM M *solves* or *decides* a decision problem D over an alphabet Σ if M stops on every input $x \in \Sigma^*$ with an answer:

- Yes (i.e., at configuration $(f, \varepsilon, 1)$) if x is a positive instance of D ;
- No (i.e., at configuration $(f, \varepsilon, 0)$) otherwise.

We say that M *partially decides* D if it decides D correctly on a subset D' of D and on $D - D'$ it either does not stop or stops with an answer *Dont Know* (i.e., stops at configuration (f, ε, \sqcup)).

3.1.2. Non-deterministic Turing machines.

DEFINITION 3.1.2. A one-tape *non-deterministic Turing machine* (NTM) M is a 5-tuple $\langle Q, \Sigma, s, f, \delta \rangle$, where Q, Σ, s, f and δ are as in the definition of deterministic TM except that $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is a multivalued function (or a binary relation).

Configurations of an NTM are the same as configurations of a deterministic TM, but the relation “yields in one step” is slightly different. We say that an NTM M , given a configuration $c_1 = (q, w, u)$, yields a configuration $c_2 = (q', w', u')$ if there exists a transition rule which transforms c_1 to c_2 . As before define the relation “yields” based on the relation “yields in one step”. Note that, according to this definition of a configuration, an NTM can yield two or more configurations in one step and exponentially many configurations in n steps. Moreover, it is allowed to yield both halting and non-halting configurations on the same input, which means we need a revised definition of acceptance. Thus, we will say that M *accepts* an input x if the initial configuration yields a halting configuration.

3.1.3. Probabilistic Turing machines. Intuitively, a *probabilistic Turing machine* is a Turing machine with a random number generator. More precisely, it is a machine with two transition functions δ_1 and δ_2 . At each step of computation, each function δ_i is used with probability $\frac{1}{2}$. Probabilistic Turing machines are similar to nondeterministic Turing machines in the sense that one configuration can yield many configurations in n steps, with the difference of how we interpret computations. For an NTM M we are interested in the question whether or not there is a sequence of choices that make M accept a certain input, whereas for the same PTM M the question is with what probability acceptance occurs.

The output of a probabilistic machine M on input x is a random variable, denoted by $M(x)$. By $\mathbf{P}(M(x) = y)$ we denote the probability for the machine M to output y on the input x . The probability space is the space of all possible outcomes of the internal coin flips of M taken with uniform probability distribution.

DEFINITION 3.1.3. Let D be a decision problem. We say that a PTM M decides D if it outputs the right answer with probability at least $2/3$.

3.2. Computational problems

In this section we briefly discuss general definitions of computational (or algorithmic) problems, size functions and stratification. At the end of the section we recall basics of the worst-case analysis of algorithms, introduce the worst-case complexity, and discuss its limitations. One of the main purposes of this section is to introduce notation and terminology.

3.2.1. Decision and search computational problems. We start with the standard definitions of decision and search computational problems, though presented in a slightly more general, than usual, form.

Let X be a finite alphabet and X^* the set of all words in the alphabet X . Sometimes subsets of X^* are called *languages* in X . A *decision problem* for a subset $L \subseteq X^*$ is the following problem:

is there an algorithm that for a given word $w \in X^*$ determines whether w belongs to L or not?

If such an algorithm exists, we call it a *decision algorithm* for L , and in this case the decision problem for L , as well as the language L , is called *decidable*. More formally, a decision problem is given by a pair $\mathcal{D} = (L, X^*)$, with $L \subseteq X^*$. We refer to words from X^* as *instances* of the decision problem \mathcal{D} . The set L is termed the *positive*, or the *Yes*, part of the problem \mathcal{D} , and the complement $\bar{L}(\mathcal{D}) = X^* - L$ is the *negative*, or the *No*, part of \mathcal{D} . If there are no decision algorithms for \mathcal{D} , then \mathcal{D} is called *undecidable*.

In practice, many decision problems appear in the *relativized* form $\mathcal{D} = (L, U)$, where $L \subseteq U \subseteq X^*$, so the set of instances of \mathcal{D} is restricted to the subset U . For example, the famous Diophantine Problem for integers (10th Hilbert's problem) asks whether a given polynomial with integer coefficients has an integer root or not. In this case polynomials are usually presented by specific words (terms) in the corresponding alphabet and there is no need to consider arbitrary words as instances. Typically, the set U is assumed to be decidable, or at least effectively enumerable.

Sometimes decision problems occur in a more general form given by a pair $\mathcal{D} = (L, U)$, where $L \subseteq U$ and U is a subset of the Cartesian product $X^* \times \dots \times X^*$ of $k \geq 1$ copies of X^* . For example, the conjugacy problem in a given group is naturally described by a set of pairs of elements of the group; or the isomorphism problem for graphs is usually given by a set of pairs of graphs, etc. By introducing an extra alphabet symbol “,” one could view a k -tuple of words $(w_1, w_2, \dots, w_k) \in (X^*)^k$ as a single word in the alphabet $X' = X \cup \{\}\$, which allows one to view the decision problems above as given, again, in the form (L, U) , with $U \subseteq (X')^*$.

A *search computational problem* can be described by a binary predicate $R(x, y) \subseteq X^* \times Y^*$, where X and Y are finite alphabets. In this case, given an input $x \in X^*$, one has to find a word $y \in Y^*$ such that $R(x, y)$ holds. For example, in the Diophantine Problem above x is a polynomial equation (or a “word” describing this equation) $E_x(y) = 0$ in a tuple of variables y , and the predicate $R(x, y)$ holds for given values of x and y if and only if these values give a solution of $E_x(y) = 0$. In what follows we always assume that the predicate $R(x, y)$ is computable.

In general, one can consider two different variations of search problems. The first one requires, for a given $x \in X^*$, to decide first whether there exists $y \in Y^*$ such that $R(x, y)$ holds, and only after that to find such y if it exists. In this case the search problem $\mathcal{D} = (R, X^* \times Y^*)$ contains the decision problem $\exists y R(x, y)$ as a subproblem. In the second variation one assumes from the beginning that for a given x the corresponding y always exists and the problem is just to find such y . The latter can be described in the relativized form by $\mathcal{D} = (R, U \times Y^*)$, where U is the “Yes” part of the decision problem $\exists y R(x, y)$.

Quite often algorithmic search problems occur as “decision problems with witnesses”. For example, given a pair of elements (x_1, x_2) of a given group G , one

has to check if they are conjugate in G or not, and if they are, one has to find a “witness” to that, i.e., one of the conjugating elements. Similarly, given two finite graphs Γ_1 and Γ_2 one has to solve the isomorphism problem for Γ_1, Γ_2 , and if the graphs are isomorphic, to find an isomorphism witnessing the solution.

If the predicate $R(x, y)$ is computable, then the second variation of the corresponding search problem is always decidable (it suffices to verify all possible inputs from Y^* one by one to find a solution if it exists). Note also that in this case the first variation is also decidable provided the decision problem $\exists y R(x, y)$ is decidable. Hence, in this situation the decidability of the search algorithmic problems follows from the decidability of the corresponding decision problems. However, in many cases our main concern is about complexity of the decision or search algorithms, so in this respect, search computational problems a priori cannot be easily reduced to decision problems.

Our final remark on descriptions of computational problems is that quite often inputs for a particular problem are not given by words in an alphabet; moreover, any representation of inputs by words, although possible, brings unnecessary complications into the picture. Furthermore, such a representation may even change the nature of the problem itself. For example, in computational problems for graphs it is possible to represent graphs by words in some alphabet, but it is not very convenient, and sometimes misleading (see [124] for details). In these cases we feel free to represent instances of the problem in a natural way, not encoding them by words. One can easily adjust all the definitions above to such representations. In the most general way, we view a computational decision problem as a pair $\mathcal{D} = (L, I)$, where $I = I_{\mathcal{D}}$ is a set of inputs for \mathcal{D} and $L = L_{\mathcal{D}}$ is a subset of I , the “Yes” part of \mathcal{D} . Similarly, a general search computational problem can be described as $\mathcal{D} = (R(x, y), I)$, where I is a subset of $I_1 \times I_2$ and $R(x, y) \subseteq I$. We always assume that the set I , as well as all its elements, allows an effective description. We do not want to go into details on this subject, but in every particular case this will be clear from the context. For example, we may discuss algorithms over matrices or polynomials with coefficients from finite fields or rational numbers, or finite graphs, or finitely presented groups, etc., without specifying any particular representations of these objects by words in a particular alphabet.

3.2.2. Size functions. In this section we discuss various ways to introduce the size, or complexity, of instances of algorithmic problems. This is part of a much bigger topic on complexity of descriptions of mathematical objects.

To study computational complexity of algorithms one has to be able to compare the resources $r_{\mathcal{A}}(x)$ spent by an algorithm \mathcal{A} on a given input x with the “size” $\text{size}(x)$ (or “complexity”) of the input. In what follows we mostly consider only one resource: the time spent by the algorithm on a given input. To get rid of inessential details coming into the picture from a particular model of computation or the way the inputs are encoded, one can consider the growth of the “resource” function $r_{\mathcal{A}} : \text{size}(x) \rightarrow r(x)$. This is where the notion of the size of inputs plays an important role and greatly affects behavior of the function $r_{\mathcal{A}}$. Usually the size of an input x depends on how the inputs are described. For example, if a natural number x is given in a unary number system, i.e., x is viewed as a word of length x in a unary alphabet $\{1\}$, say $x = 11\dots 1$, then it is natural to assume that the size of x is the length of the word representing x , i.e., is x itself. However, if the natural number x is given, say, in the binary or decimal representation, then the

size of x (which is the length of the corresponding representation) is exponentially smaller (about $\log x$), so the same algorithm \mathcal{A} may have quite different resource functions depending on the choice of the size function.

The choice of the size function depends, of course, on the problem \mathcal{D} . There are two principle approaches to define size functions on a set I . In the first one, the size of an input $x \in I$ is the *descriptive complexity* $d(x)$ of x , which is the length of the minimal description of x among all natural representations of x of a fixed type. For example, the length $|w|$ of a word w in a given alphabet is usually viewed as the size of w . However, there are other natural size functions, for instance, the size of an element g of a finitely generated group G could be the length of a shortest product of elements and their inverses (from a fixed finite generating set of G) which is equal to g . One of the most intriguing size functions in this class comes from the so-called Kolmogorov complexity (see [172] for details). In the second approach, the size of an element $x \in I$ is taken to be the time required for a given generating procedure to generate x (*production complexity*). For example, when producing keys for a cryptoscheme, it is natural to view the time spent on generating a key x as the size $c(x)$ of x . In this case, the computational security of the scheme may be based on the amount of recourses required to break x relative to the size $c(x)$. It could easily be that the descriptive complexity $d(x)$ of x is small but the computational complexity $c(x)$ (with respect to a given generating procedure) is large.

In general, a *size* (or *complexity*) *function* on a set I is an arbitrary non-negative integral (or real) function $s : I \rightarrow \mathbb{N}^+$ (or $s : I \rightarrow \mathbb{R}^+$) that satisfies the following conditions:

- C1) For every $n \in \mathbb{N}$ the preimage $s^{-1}(n)$ is either a finite subset of I or, in the case where I is equipped with a measure, a measurable subset of I . If s is a size function with values in \mathbb{R}^+ and I is equipped with a measure μ , then we require (if not said otherwise) that s is μ -measurable.
- C2) For an element $x \in I$ given in a fixed representation, one can effectively compute $s(x)$.

Note that it makes sense to only consider size functions on I which are “natural” in the context of a problem at hand. This is an important principle that we do not formally include in the conditions above because of the difficulties with formalization. For an algorithmic problem \mathcal{D} , by $s_{\mathcal{D}}$ we denote the given size function on the set of instances I (if it exists).

There is another very interesting way to define size functions on sets of inputs of algorithms, which is not yet developed enough. We briefly sketch the main idea here.

Let \mathcal{C} be a class of decision or search algorithms solving a computational problem $\mathcal{D} = (L, I)$ that use different strategies but employ more or less similar tools. Let A_{opt} be a non-deterministic algorithm, which is based on the same tools as algorithms from \mathcal{C} are, but which is using an oracle that provides an optimal strategy at each step of computation. One can define the size $s_{opt}(x)$ of $x \in I$ as the time spent by the algorithm A_{opt} on the input x . Then for any particular algorithm $A \in \mathcal{C}$ the resource function r_A will allow one to evaluate the performance of the algorithm A in comparison with the optimal (in the class \mathcal{C}) “ideal” algorithm A_{opt} . This approach proved to be very useful, for example, in the study of algorithmic complexity of the famous Whitehead problem from group theory and topology (see [209]).

3.2.3. Stratification. In this section we introduce a notion of *stratification* relative to a given size function and some related terminology. Stratification provides important tools to study asymptotic behavior of subsets of inputs.

Let $s : I \rightarrow \mathbb{N}$ be a size function on a set of inputs I , satisfying the conditions C1), C2) from Section 3.2.2. For a given $n \in \mathbb{N}$ the set of elements of size n in I ,

$$I_n = \{x \in I \mid s(x) = n\}$$

is called the *sphere* of radius n (or n -stratum), whereas the set

$$B_n(I) = \bigcup_{k=1}^n I_k = \{x \in I \mid s(x) \leq n\}$$

is called the *ball* of radius n .

The partition

$$(2) \quad I = \bigcup_{k=1}^{\infty} I_k$$

is called *size stratification* of I , and the decomposition

$$I = \bigcup_{k=1}^{\infty} B_k(I)$$

is the *size volume decomposition*. The converse also holds, i.e., every partition (2) induces a size function on I such that $x \in I$ has size k if and only if $x \in I_k$. In this case the condition C1) holds if and only if the partition (2) is computable, whereas C2) holds if and only if the sets I_k are finite (or measurable, if the set I comes equipped with a measure).

Size stratifications and volume decompositions allow one to study asymptotic behavior of various subsets of I , as well as some functions defined on some subsets of I . For example, if the size function $s : I \rightarrow \mathbb{N}$ is such that for any $n \in \mathbb{N}$ the set I_n is finite (see condition C1), then for a given subset $R \subseteq I$ the function $n \rightarrow |R \cap I_n|$ is called the *growth* function of R (relative to the given size stratification), while the function

$$n \rightarrow \rho_n(R) = \frac{|R \cap I_n|}{|I_n|}$$

is called the *frequency* function of R . The asymptotic behavior of R can be seen via its *spherical asymptotic density*, which is defined as the following limit (if it exists):

$$\rho(R) = \lim_{n \rightarrow \infty} \rho_n(R).$$

Similarly, one can define the *volume asymptotic density* of R as the limit (if it exists) of the *volume frequencies*:

$$\rho^*(R) = \lim_{n \rightarrow \infty} \rho_n^*(R),$$

where

$$\rho_n^*(R) = \frac{|R \cap B_n(I)|}{|B_n(I)|}.$$

One can also define the density functions above using \limsup rather than \lim . We will have more to say about this in Section 10.1.2.

These spherical and volume stratifications play an important role in the study of complexity of algorithms and algorithmic problems.

3.2.4. Reductions and complete problems. In this section we discuss what it means when we say that one problem is as hard as another. A notion of *reduction* is used to define this concept. Intuitively, if a problem D_1 is reducible to a problem D_2 , then via some “reduction procedure” a decision algorithm for D_2 gives a decision algorithm to D_1 . This allows one to estimate the hardness of the problem D_1 through the hardness of D_2 and through the hardness of the reduction procedure.

One of the most general type of reductions is the so-called *Turing reduction* (see Section 3.2.6). It comes from the classical recursion theory and fits well for studying undecidable problems. Below we give a general description of one of the most common reductions, *many-to-one reduction* (see Section 3.2.5).

Let F be a set of functions from \mathbb{N} to \mathbb{N} , and D_1 and D_2 some decision problems (say, subsets of \mathbb{N}). The problem D_1 is called *reducible* to D_2 under F if there exists a function $f \in F$ satisfying

$$x \in D_1 \Leftrightarrow f(x) \in D_2.$$

In this case we write $D_1 \leq_F D_2$. It is natural to assume that the set F contains all identity functions and hence any problem D is reducible to itself. Moreover, usually the set F is closed under compositions which implies that if $D_1 \leq_F D_2$ and $D_2 \leq_F D_3$, then $D_1 \leq_F D_3$. Thus, \leq_F is reflexive and transitive.

Now let S be a set of decision problems and \leq_F a reducibility relation. We say that S is *closed under reductions* \leq_F if for any problem D_1 and a problem $D_2 \in S$, one has

$$D_1 \leq_F D_2 \Leftrightarrow D_1 \in S.$$

A problem C is called *hard* for S if for any $D \in S$ we have $D \leq_F C$. A problem C is called *complete* for S if it is hard for S and $C \in S$.

3.2.5. Many-to-one reductions. Let D_1 and D_2 be decision problems. A recursive function $f : D_1 \rightarrow D_2$ is called a *many-to-one reduction* of D_1 to D_2 if $x \in D_1$ if and only if $f(x) \in D_2$. In this case we say that D_1 is many-to-one reducible or *m-reducible* to D_2 and write $D_1 \leq_m D_2$. If f is an injective many-to-one reduction, then we say that D_1 is one-to-one reducible to D_2 and write $D_1 \leq_1 D_2$.

Many-to-one reductions are a special case and a weaker form of Turing reductions. With many-to-one reductions only one invocation of the oracle is allowed, and only at the end.

Recall that a class S of problems is closed under many-to-one reducibility if there are no many-to-one reductions from a problem in S to a problem outside S . If a class S is closed under many-to-one reducibility, then many-to-one reductions can be used to show that a problem is in S by reducing a problem in S to it.

The class of all decision problems has no complete problem with respect to many-to-one reductions.

3.2.6. Turing reductions. In this section we discuss the most general type of reductions called *Turing reductions*. Turing reductions are very important in computability theory and in providing theoretical grounds for more specific types of reductions, but like many-to-one reductions discussed in the previous section, they are of limited practical significance.

We say that a problem D_1 is *Turing reducible* to a problem D_2 and write $D_1 \leq_T D_2$ if D_1 is computable by a Turing machine with an oracle for D_2 . The

complexity class of decision problems solvable by an algorithm in class A with an oracle for a problem in class B is denoted by A^B . For example, the class of problems solvable in polynomial time by a deterministic Turing machine with an oracle for a problem in **NP** is **P^{NP}**. (This is also the class of problems reducible by a polynomial-time Turing reduction to a problem in **NP**.)

It is possible to postulate the existence of an oracle which computes a non-computable function, such as an answer to the halting problem. A machine with an oracle of this sort is called a *hypercomputer*. Interestingly, the halting paradox still applies to such machines; that is, although they can determine whether particular Turing machines will halt on particular inputs, they cannot determine whether machines with equivalent halting oracles will themselves halt. This fact creates a hierarchy of machines, called the *arithmetical hierarchy*, each with a more powerful halting oracle and an even harder halting problem.

3.3. The worst case complexity

3.3.1. Complexity classes. We follow the book *Computational Complexity* by C. Papadimtriou [219] for our conventions on computational complexity. A *complexity class* is determined by specifying a *model of computation* (which for us is almost always a Turing machine), a *mode of computation* (e.g. deterministic or non-deterministic), *resources* to be controlled (e.g. time or space) and *bounds* for each controlled resource, which is a function f from non-negative integers to non-negative integers satisfying certain properties as discussed below. The complexity class is now defined as the set of all languages decided by an appropriate Turing machine M operating in the appropriate mode and such that, for any input x , M uses at most $f(|x|)$ units of the specified resource. There are some restrictions on the bound function f . Speaking informally, we want to exclude those functions that require more resources to be computed than to read an input n and an output $f(n)$.

DEFINITION 3.3.1. Let f be a function from non-negative integers to non-negative integers. We say that f is a proper complexity function if the following conditions are satisfied:

- (1) f is non-decreasing, i.e., for any $n \in \mathbb{N}$ $f(n+1) \geq f(n)$.
- (2) There exists a multitape Turing machine M_f such that for any n and for any input x of length n , M computes a string $0^{f(|x|)}$ in time $T_M(x) = O(n + f(n))$ and space $S_M(x) = O(f(n))$.

The class of proper complexity bounds is very extensive and excludes mainly pathological cases of functions. In particular, it contains all polynomial functions, functions \sqrt{n} , $n!$, and $\log n$. Moreover, with any functions f and g it contains the functions $f + g$, $f \cdot g$, and 2^f .

For a proper complexity function f , we define **TIME**(f) to be the class of all decision problems decidable by some deterministic Turing machine within time $f(n)$. Then, let **NTIME**(f) be the class of all decision problems decidable by some non-deterministic Turing machine within time $f(n)$. Similarly, we can define the complexity classes **SPACE**(f) (deterministic space) and **NSPACE**(f) (non-deterministic space).

Often classes are defined not for particular complexity functions but for parameterized families of functions. For instance, the two most important classes of

decision problems, **P** and **NP**, are defined as follows:

$$\begin{aligned}\mathbf{P} &= \bigcup_{k=1}^{\infty} \mathbf{TIME}(n^k), \\ \mathbf{NP} &= \bigcup_{k=1}^{\infty} \mathbf{NTIME}(n^k).\end{aligned}$$

Other important classes are **PSPACE** = $\bigcup_{k=1}^{\infty} \mathbf{SPACE}(n^k)$ (polynomial deterministic space) and **NPSPACE** = $\bigcup_{k=1}^{\infty} \mathbf{NSPACE}(n^k)$ (polynomial non-deterministic space), **EXP** = $\bigcup_{k=1}^{\infty} \mathbf{TIME}((2^n)^k)$ (exponential deterministic time).

As we have mentioned above, the principal computational models for us are Turing machines (in all their variations). However, we sometimes may describe them not by Turing machine programs, but rather less formally, as they appear in mathematics. Moreover, we may use other sequential models of computation, keeping in mind that we can always convert them into Turing machines, if needed. We refer to all these different descriptions and different models of computation as *algorithms*.

The complexity classes defined above are the so-called *worst-case complexity* classes.

3.3.2. Class NP. It is a famous open problem whether **P** = **NP** or not. There are several equivalent definitions of **NP**. According to our definition, the language $L \subseteq \{0, 1\}^*$ belongs to **NP** if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0, 1\}^*$,

$$x \in L \text{ if and only if } \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1.$$

If $x \in L$ and $u \in \{0, 1\}^{p(|x|)}$ satisfies $M(x, u) = 1$ then we say that u is a *certificate* for x (with respect to the language L and machine M).

DEFINITION 3.3.2. A Boolean formula φ over variables u_1, \dots, u_n is in *conjunctive normal form* (CNF) if it is of the form

$$\wedge(\vee v_{i_j}),$$

where each v_{i_j} is a literal of φ ; in other words, either a variable u_k or its negation \bar{u}_k . The terms $\vee v_{i_j}$ are called *clauses*. If all clauses contain at most k literals, the formula is called k -CNF.

Several problems in **NP**:

- (1) **Satisfiability problem, or SAT:** Given a Boolean formula φ in conjunctive normal form over variables u_1, \dots, u_n , determine whether or not there is a satisfying assignment for φ .
- (2) **Three satisfiability problem, or 3SAT:** Given a Boolean formula φ in conjunctive normal form over variables u_1, \dots, u_n , where each conjunct contains up to 3 variables, determine whether or not there is a satisfying assignment for φ .
- (3) **Subset sum problem:** Given a list of n numbers a_1, \dots, a_n and a number A , decide whether or not there is a subset of these numbers that sums up to A . The certificate is the list of members in such a subset.

- (4) **Graph isomorphism:** Given two $n \times n$ adjacency matrices M_1, M_2 , decide whether or not M_1 and M_2 define the same graph up to renaming of vertices. The certificate is a permutation $\pi \in S_n$ such that M_1 is equal to M_2 after reordering M_1 's entries according to π .
- (5) **Linear programming:** Given a list of m linear inequalities with rational coefficients over n variables u_1, \dots, u_n (a linear inequality has the form $a_1u_1 + a_2u_2 + \dots + a_nu_n \leq b$ for some coefficients a_1, \dots, a_n, b), decide whether or not there is an assignment of rational numbers to the variables u_1, \dots, u_n that satisfies all the inequalities. The certificate is the assignment.
- (6) **Integer programming:** Given a list of m linear inequalities with rational coefficients over n variables u_1, \dots, u_n decide whether or not there is an assignment of integers to u_1, \dots, u_n that satisfies all the inequalities. The certificate is the assignment.
- (7) **Travelling salesperson:** Given a set of n nodes, $\binom{n}{2}$ numbers $d_{i,j}$ denoting the distances between all pairs of nodes, and a number k , decide whether or not there is a closed circuit (i.e., a “salesperson tour”) that visits every node exactly once and has total length at most k . The certificate is the sequence of nodes in the tour.
- (8) **Bounded halting problem:** Let M be a non-deterministic Turing machine with binary input alphabet. The bounded halting problem for M (denoted by $H(M)$) is the following algorithmic problem. For a positive integer n and a binary string x decide whether or not there is a halting computation of M on x within at most n steps. The certificate is the halting execution flow (sequence of states of M).
- (9) **Post correspondence problem:** Given a nonempty list $L = ((u_1, v_1), \dots, (u_s, v_s))$ of pairs of binary strings and a positive integer n determine whether or not there is a tuple $i = (i_1, \dots, i_k)$ in $\{1, \dots, s\}^k$, where $k \leq n$, such that

$$u_{i_1}u_{i_2}\dots u_{i_k} = v_{i_1}v_{i_2}\dots v_{i_k}.$$

The certificate is the tuple i .

3.3.3. Polynomial-time many-to-one reductions and class NP. Many-to-one reductions are often subjected to resource restrictions, for example, that the reduction function is computable in polynomial time or logarithmic space. If not chosen carefully, the whole hierarchy of problems in the class can collapse to just one problem. Given decision problems D_1 and D_2 and an algorithm A which solves instances of D_2 , we can use a many-to-one reduction from D_1 to D_2 to solve instances of D_2 in:

- the time needed for A plus the time needed for the reduction;
- the maximum of the space needed for A and the space needed for the reduction.

Recall that **P** is the class of decision problems that can be solved in polynomial time by a deterministic Turing machine and **NP** is the class of decision problems that can be solved in polynomial time by a non-deterministic Turing machine. Clearly **P** \subseteq **NP**. It is a famous open question (see e.g. [140]) whether **NP** \subseteq **P** or not.

DEFINITION 3.3.3. Let D_1 and D_2 be decision problems. We say that $f : D_1 \rightarrow D_2$ *Ptime reduces* D_1 to D_2 and write $D_1 \leq_P D_2$ if

- f is polynomial time computable;
- $x \in D_1$ if and only if $f(x) \in D_2$.

We say that a Ptime reduction f is *size-preserving* if

- $|x_1| < |x_2|$ if and only if $|f(x_1)| < |f(x_2)|$.

Recall that 3SAT is the following decision problem. Given a Boolean expression in conjunctive normal form, where each clause contains at most 3 variables, find out whether or not it is satisfiable. It is easy to check that 3SAT belongs to **NP**. The following theorem is a classical result.

THEOREM 3.3.4. *The following is true:*

- 1) **NP** is closed under Ptime reductions.
- 2) If f is a Ptime reduction from D_1 to D_2 and M is a Turing machine solving D_2 in polynomial time, then the composition of f and M solves D_1 in polynomial time.
- 3) 3SAT is complete in **NP** with respect to Ptime reductions.

It follows from Theorem 3.3.4 that to prove (or disprove) that $\mathbf{NP} \subseteq \mathbf{P}$ one does not have to check all problems from **NP**; it is sufficient to show that $C \in \mathbf{P}$ (or $C \notin \mathbf{P}$) for any **NP**-complete problem C .

3.3.4. NP-complete problems. In Section 3.2.4 we defined a notion of a hard and complete problem for a class of decision problems. In this section we use it to define **NP**-complete problems. Recall that a decision problem $L \subseteq \{0, 1\}^*$ belongs to the class **NP** if membership in L has a polynomial-time verifiable certificate (witness). Now a problem D is called **NP-hard** if for every problem $D' \in \mathbf{NP}$ there exists a polynomial-time reduction of D' to D . A problem is said to be **NP-complete** if it belongs to **NP** and is **NP-hard**.

THEOREM 3.3.5. *There exists a nondeterministic TM U such that $H(U)$ is **NP**-complete.*

Proof. (A sketch.) As we mentioned in Section 3.3.2, for any polynomial-time TM U , one has $H(U) \in \mathbf{NP}$. On the other hand, $D \in \mathbf{NP}$ if and only if there exists a polynomial-time NTM M deciding D . Let $p(n)$ be the time-complexity of M . Fix an arbitrary problem D from **NP**. Let U be a universal NTM (which simulates execution flow of any TM M on any input x) such that:

- (1) U takes inputs of the form $1^m 0x$, where m is a Gödel number of a TM M to be simulated and x is an input for M .
- (2) U is an efficient universal Turing machine, i.e., there exists a polynomial $q(n)$ for U such that M stops on x in k steps if and only if U stops on $1^{m_M} 0x$ in $q(k)$ steps.

Clearly, a function f which maps an input x for M to the input $(q(p(|x|)), 1^m 0x)$ for $H(U)$ is a polynomial-time reduction. Thus, $H(U)$ is **NP-hard**. ■

Let M be a TM. We say that M is oblivious if its head movement depends only on the length of input as a function of steps. It is not difficult to see that for any Turing machine M with time complexity $T(n)$ there exists an oblivious TM M' computing the same function as M does with time complexity $T^2(n)$.

To prove the next theorem we need to introduce a notion of a snapshot for execution of a Turing machine. Assume that $M = \langle Q, \Sigma, s, f, \delta \rangle$ is a TM with 2 tapes. The first tape is read-only and is referred to as an *input tape*, and the second tape is in the read-write mode and is referred to as the *working tape*. A *snapshot* of M 's execution on input y at step i is a triple $(a, b, q) \in \Sigma \times \Sigma \times Q$, where a and b are the symbols observed by the the M 's head and q is the current state.

THEOREM 3.3.6. *SAT is **NP**-complete*

Proof. SAT clearly belongs to **NP**, hence it remains to show that it is **NP**-hard. Let D be an arbitrary **NP** problem. Our goal is to construct a polynomial time reduction f which maps instances x of D to CNF formulas φ_x such that $x \in D$ if and only if φ_x is satisfiable. By definition, $D \in \mathbf{NP}$ if and only if there exists a polynomial $p(n)$ and a polynomial-time 2-tape TM $M = \langle Q, \Sigma, s, f, \delta \rangle$ such that for every $x \in \{0, 1\}^*$,

$$x \in D \text{ if and only if } \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1.$$

As mentioned above, we may assume that M is oblivious (i.e., its head movement does not depend on the contents of the input tape, but only on the length of the input). The advantage of this property is that there are polynomial-time computable functions

- (1) $inputpos(i)$ – the location of the input tape head at step i ;
- (2) $prev(i)$ – the last step before i that M visited the same location on the work tape.

Each snapshot of M 's execution can be encoded by a binary string of length c , where c is a constant depending on parameters of M , but independent of the input length.

For every $m \in \mathbb{N}$ and $y \in \{0, 1\}^m$, the snapshot of M 's execution on the input y at the i th step depends on its state at the $(i - 1)$ th step and on the contents of the current cells of its input and work tapes. Thus, if we denote the encoding of the i th snapshot as a string of length c by z_i , then z_i is a function of $z_{i-1}, y_{inputpos(i)}$, and $z_{prev(i)}$, i.e.,

$$z_i = F(z_{i-1}, y_{inputpos(i)}, z_{prev(i)}),$$

where F is some function that maps $\{0, 1\}^{2c+1}$ to $\{0, 1\}^c$.

Let $n \in \mathbb{N}$ and $x \in \{0, 1\}^n$. We need to construct a CNF φ_x such that $x \in D$ if and only if $\varphi_x \in \text{SAT}$. Recall that $x \in D$ if and only if there exists $u \in \{0, 1\}^{p(n)}$ such that $M(y) = 1$, where $y = x \circ u$. Since the sequence of snapshots of M 's execution completely determines its outcome, this happens if and only if there exists a string $y \in \{0, 1\}^{n+p(n)}$ and a sequence of strings $z_1, \dots, z_T \in \{0, 1\}^c$, where $T = T(n)$ is the number of steps M takes on input of length $n + p(n)$ satisfying the following 4 conditions:

- (1) The first n bits of y are the same as in x .
- (2) The string z_1 encodes the initial snapshot of M .
- (3) For every $i = 2, \dots, T$, $z_i = F(z_{i-1}, y_{inputpos(i)}, z_{prev(i)})$.
- (4) The last string z_T encodes a snapshot in which the machine halts and outputs 1.

The formula φ_x takes variables $y \in \{0, 1\}^{n+p(n)}$ and $z \in \{0, 1\}^{cT}$ and verifies that y, z satisfy all of these four conditions. Clearly, $x \in D$ if and only if $\varphi_x \in \text{SAT}$, so it remains to show that we can express φ_x as a polynomial-size CNF formula.

Condition (1) can be expressed as a CNF formula of size $2n$. Conditions (2) and (4) each depend on c variables and hence can be expressed by a CNF formula of size $c2^c$. Condition (3) is an AND of T conditions each depending on at most $3c + 1$ variables, hence it can be expressed as a CNF formula of size at most $T(3c + 1)2^{3c+1}$. Hence, AND of all these conditions can be expressed as a CNF formula of size $d(n + T)$, where d is a constant depending only on M . Moreover, this CNF formula can be computed in polynomial time in the running time of M . ■

3.3.5. Deficiency of the worst case complexity. The worst-case complexity of algorithms tells something about resources spent by a given algorithm \mathcal{A} on the “worst possible” inputs for \mathcal{A} . Unfortunately, when the worst-case inputs are sparse, this type of complexity tells nothing about the actual behavior of the algorithm on “most” or “most typical” (*generic*) inputs. However, what really counts in many practical applications is the behavior of algorithms on typical, most frequently occurring, inputs. For example, Dantzig’s simplex algorithm for linear programming problems is used thousands of times daily and in practice almost always works quickly. It was shown by V. Klee and G. Minty [159] that there are some inputs where the simplex algorithm requires exponential time to finish computation, so it has exponential worst-case time complexity. In [156] Khachiyan came up with a new ingenious algorithm for linear programming problems, that has polynomial time complexity. Nevertheless, at present Dantzig’s algorithm continues to be most widely used in applications. The reason is that a “generic”, or “random”, linear programming problem is not “special”, and the simplex algorithm works quickly. Mathematical justification of this phenomenon has been found independently by Vershik and Sporyshev [280] and Smale [260]. They showed that on a set of problems of measure one Dantzig’s simplex algorithm has linear time complexity.

Modern cryptography is another area where computational complexity plays a crucial role because modern security assumptions in cryptography require analysis of behavior of algorithms on random inputs. The basic computational problem, which is in the core of a given cryptoscheme, must be hard on *most* inputs, to make it difficult for an attacker to crack the system. In this situation the worst-case analysis of algorithms is irrelevant.

Observations of this type have led to the development of new types of complexity measure, where computational problems come equipped with a fixed distribution on the set of inputs (so-called *distributional*, or *randomized*, computational problems). This setting allows one to describe the behavior of algorithms either on average or on “most” inputs. We discuss these complexity measures in more detail in Part 3 of this book.

Part 2

Non-commutative Cryptography

The object of this part of the book is to explore how to use the complexity of non-commutative groups in *public-key cryptography*.

The idea of using the complexity of infinite, non-abelian groups in cryptography goes back to Wagner and Magyarik [180] who in 1985 devised a public-key protocol based on the unsolvability of the word problem for finitely presented groups (or so they thought). Their protocol now looks somewhat naive, but it was pioneering. More recently, there has been an increased interest in applications of non-abelian group theory to cryptography (see for example [5, 161, 257]). Most suggested protocols are based on *search problems* which are variants of more traditional *decision problems* of combinatorial group theory. Protocols based on search problems fit in with the general paradigm of a public-key protocol based on a one-way function (see above). We therefore dub the relevant area of cryptography *canonical cryptography* and explore it in Chapter 4 of our book.

On the other hand, employing decision problems in public-key cryptography allows one to depart from the canonical paradigm and construct cryptographic protocols with new properties, impossible in the canonical model. In particular, such protocols can be secure against some “brute force” attacks by a computationally unbounded adversary. There is a price to pay for that, but the price is reasonable: a legitimate receiver decrypts correctly with probability that can be made very close to 1, but not equal to 1. We discuss this and other new ideas in Chapter 15.

In a separate Chapter 5 we describe groups that can be used as platforms for cryptographic protocols from Chapters 4 and 15 of this book. These include braid groups, groups of matrices, small cancellation groups, and others.

CHAPTER 4

Canonical Non-commutative Cryptography

In this and the following chapter, we discuss various cryptographic primitives that use non-commutative (semi)groups as platforms, but at the same time we do not depart from the canonical paradigm of a public-key protocol based on a trapdoor one-way function. We include here the ground-breaking Anshel-Anshel-Goldfeld protocol [5] as well as protocols that are closer in spirit to classical protocols based on commutative (semi)groups.

4.1. Protocols based on the conjugacy search problem

Let G be a group with solvable word problem. For $w, a \in G$, the notation w^a stands for $a^{-1}wa$. Recall that the *conjugacy problem* (or *conjugacy decision problem*) for G is: given two elements $u, v \in G$, find out whether there is $x \in G$ such that $u^x = v$. On the other hand, the *conjugacy search problem* (sometimes also called the *conjugacy witness problem*) is: given two elements $a, b \in G$ and the information that $u^x = v$ for some $x \in G$, find at least one particular element x like that.

The conjugacy decision problem is of great interest in group theory. In contrast, the conjugacy search problem is of interest in complexity theory, but of no interest in group theory. Indeed, if you know that u is conjugate to v , you can just go over words of the form u^x and compare them to v one at a time, until you get a match. (We implicitly use here an obvious fact that a group with solvable conjugacy problem also has solvable word problem.) This straightforward algorithm is at least exponential-time in the length of v , and therefore is considered infeasible for practical purposes.

Thus, if no other algorithm is known for the conjugacy search problem in a group G , it is not unreasonable to claim that $x \rightarrow u^x$ is a one-way function and try to build a (public-key) cryptographic protocol on that.

We start with a simple protocol, due to Ko, Lee et al. [161].

- (1) An element $w \in G$ is published.
- (2) Alice picks a private $a \in G$ and sends w^a to Bob.
- (3) Bob picks a private $b \in G$ and sends w^b to Alice.
- (4) Alice computes $K_A = (w^b)^a = w^{ba}$, and Bob computes $K_B = (w^a)^b = w^{ab}$.

If a and b are chosen from a pool of commuting elements of the group G , then $ab = ba$, and therefore, Alice and Bob get a common private key $K_B = w^{ab} = w^{ba} = K_A$. Typically, there are two public subgroups A and B of the group G , given by their (finite) generating sets, such that $ab = ba$ for any $a \in A, b \in B$.

In the paper [161], the platform group G was the braid group B_n which has some natural commuting subgroups. Selecting a suitable platform group for the

above protocol is a very nontrivial matter; some requirements on such a group were put forward in [247]:

- (P0) The group has to be well known. More precisely, the conjugacy search problem in the group either has to be well studied or can be reduced to a well-known problem (perhaps, in some other area of mathematics).

This property, although not mathematical, appears to be mandatory if we want our cryptographic product to be used in real life. We note in passing that this property already narrows down the list of candidates quite a bit.

- (P1) The word problem in G should have a fast (at most quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable “normal form” for elements of G .

This is required for an efficient common key extraction by legitimate parties in a key establishment protocol, or for the verification step in an authentication protocol, etc.

- (P2) The conjugacy search problem should *not* have an efficient solution by a deterministic algorithm.

We point out here that *proving* a group to have (P2) should be extremely difficult, if not impossible. This is, literally, a million-dollar problem (see [140]). The property (P2) should therefore be considered in conjunction with (P0), i.e., the only realistic evidence of a group G having the property (P2) can be the fact that sufficiently many people have been studying the conjugacy search problem in G over a sufficiently long time.

The next property is somewhat informal, but it is of great importance for practical implementations:

- (P3) There should be a way to disguise elements of G so that it would be impossible to recover x from $x^{-1}wx$ just by inspection.

One way to achieve this is to have a *normal form* for elements of G , which usually means that there is an algorithm that transforms any input u_{in} , which is a word in the generators of G , to an output u_{out} , which is another word in the generators of G , such that $u_{in} = u_{out}$ in the group G , but this is hard to detect by inspection.

In the absence of a normal form, say if G is just given by means of generators and relators without any additional information about properties of G , then at least some of these relators should be very short.

- (P4) G should be a group of super-polynomial (i.e., exponential or “intermediate”) growth. This means that the number of elements of length n in G should grow faster than any polynomial in n ; this is needed to prevent attacks by complete exhaustion of the key space. Here “length n ” is typically just the length of a word representing a group element, but in a more general situation, this could be the length of some other description, i.e., “information complexity”.

There are groups that have (P1), (P4), most likely have (P2), and to a reasonable extent have (P3). These are groups with solvable word problem, but unsolvable conjugacy problem (see e.g. [200]). However, the conjugacy search problem is of no independent interest in group theory, as we have mentioned in the beginning of this section. Group theorists therefore did not bother to study the conjugacy

search problem in these groups once it had been proved that the conjugacy decision problem is algorithmically unsolvable. It would probably make sense to reconsider these groups now.

4.2. Protocols based on the decomposition problem

One of the natural ramifications of the conjugacy search problem is the following *decomposition search problem*:

Given two elements w and w' of a group G , find two elements $x \in A$ and $y \in B$ that would belong to given subsets (usually subgroups) $A, B \subseteq G$ and satisfy $x \cdot w \cdot y = w'$, provided at least one such pair of elements exists.

We note that if in the above problem $A = B$ is a subgroup, then this problem is also known as the *double coset problem*.

We also note that *some* x and y satisfying the equality $x \cdot w \cdot y = w'$ always exist (e.g. $x = 1$, $y = w^{-1}w'$), so the point is to have them satisfy the conditions $x \in A$ and $y \in B$. We therefore will not usually refer to this problem as a *subgroup-restricted* decomposition search problem because it is always going to be subgroup-restricted; otherwise it does not make much sense. We also note that the most commonly considered special case of the decomposition search problem so far is where $A = B$.

We are going to show in Section 4.6 that solving the conjugacy search problem is unnecessary for an adversary to get the common secret key in the Ko-Lee (or any similar) protocol (see our Section 4.1); it is sufficient to solve a seemingly easier decomposition search problem. This was mentioned, in passing, in the paper [161], but the significance of this observation was downplayed there.

We note that the condition $x, y \in A$ may not be easy to verify for some subsets A ; the corresponding problem is known as the membership (decision) problem. The authors of [161] do not address this problem; instead they mention, in justice, that if one uses a “brute force” attack by simply going over elements of A one at a time, the above condition will be satisfied automatically. This however may not be the case with other, more practical, attacks.

We also note that the conjugacy search problem is a special case of the decomposition problem where w' is conjugate to w and $x = y^{-1}$. The claim that the decomposition problem should be easier than the conjugacy search problem is intuitively clear since it is generally easier to solve an equation with two unknowns than a special case of the same equation with just one unknown. We admit however that there might be exceptions to this general rule.

Now we give a formal description of a typical protocol based on the decomposition problem. There is a public group G , a public element $w \in G$, and two public subgroups $A, B \subseteq G$ commuting elementwise, i.e., $ab = ba$ for any $a \in A, b \in B$.

- (1) Alice randomly selects private elements $a_1, a_2 \in A$. Then she sends the element $a_1 w a_2$ to Bob.
- (2) Bob randomly selects private elements $b_1, b_2 \in B$. Then he sends the element $b_1 w b_2$ to Alice.
- (3) Alice computes $K_A = a_1 b_1 w b_2 a_2$, and Bob computes $K_B = b_2 a_1 w b_1 a_2$. Since $a_i b_i = b_i a_i$ in G , one has $K_A = K_B = K$ (as an element of G), which is now Alice’s and Bob’s common secret key.

We now discuss several modifications of the above protocol.

4.2.1. “Twisted” protocol. This idea is due to Shpilrain and Ushakov [250]; the following modification of the above protocol appears to be more secure (at least for some choices of the platform group) against so-called “length based” attacks (see e.g. [87], [86], [133], [136]), according to computer experiments. Again, there is a public group G and two public subgroups $A, B \leq G$ commuting elementwise.

- (1) Alice randomly selects private elements $a_1 \in A$ and $b_1 \in B$. Then she sends the element a_1wb_1 to Bob.
- (2) Bob randomly selects private elements $b_2 \in B$ and $a_2 \in A$. Then he sends the element b_2wa_2 to Alice.
- (3) Alice computes $K_A = a_1b_2wa_2b_1 = b_2a_1wb_1a_2$, and Bob computes $K_B = b_2a_1wb_1a_2$. Since $a_i b_i = b_i a_i$ in G , one has $K_A = K_B = K$ (as an element of G), which is now Alice’s and Bob’s common secret key.

The security of this protocol is apparently based on a more general version of the decomposition search problem than that given in the beginning of the previous subsection:

Given two elements w, w' and two subgroups A, B of a group G , find two elements $x \in A$ and $y \in B$ such that $x \cdot w \cdot y = w'$, provided at least one such pair of elements exists.

4.2.2. Hiding one of the subgroups. This idea, too, is due to Shpilrain and Ushakov [251]. First we give a sketch of the idea.

Let G be a group and $g \in G$. Denote by $C_G(g)$ the centralizer of g in G , i.e., the set of elements $h \in G$ such that $hg = gh$. For $S = \{g_1, \dots, g_k\} \subseteq G$, $C_G(g_1, \dots, g_k)$ denotes the centralizer of S in G , which is the intersection of the centralizers $C_G(g_i)$, $i = 1, \dots, k$.

Now, given a public $w \in G$, Alice privately selects $a_1 \in G$ and publishes a subgroup $B \subseteq C_G(a_1)$ (we tacitly assume here that B can be computed efficiently). Similarly, Bob privately selects $b_2 \in G$ and publishes a subgroup $A \subseteq C_G(b_2)$. Alice then selects $a_2 \in A$ and sends $w_1 = a_1wa_2$ to Bob, while Bob selects $b_1 \in B$ and sends $w_2 = b_1wb_2$ to Alice.

Thus, in the first transmission, say, the adversary faces the problem of finding a_1, a_2 such that $w_1 = a_1wa_2$, where $a_2 \in A$, but there is no explicit indication of where to choose a_1 from. Therefore, before arranging something like a length based attack in this case, the adversary would have to compute generators of the centralizer $C_G(B)$ first (because $a_1 \in C_G(B)$), which is usually a hard problem by itself since it basically amounts to finding the intersection of the centralizers of individual elements, and finding (the generators of) the intersection of subgroups is a notoriously difficult problem for most groups considered in combinatorial group theory.

Now we give a formal description of the protocol from [251]. As usual, there is a public group G , and let $w \in G$ be public, too.

- (1) Alice chooses an element $a_1 \in G$, chooses a subgroup of $C_G(a_1)$, and publishes its generators $A = \{\alpha_1, \dots, \alpha_k\}$.
- (2) Bob chooses an element $b_2 \in G$, chooses a subgroup of $C_G(b_2)$, and publishes its generators $B = \{\beta_1, \dots, \beta_m\}$.
- (3) Alice chooses a random element a_2 from $\langle \beta_1, \dots, \beta_m \rangle$ and sends $P_A = a_1wa_2$ to Bob.

- (4) Bob chooses a random element b_1 from $\langle \alpha_1, \dots, \alpha_k \rangle$ and sends $P_B = b_1 w b_2$ to Alice.
- (5) Alice computes $K_A = a_1 P_B a_2$.
- (6) Bob computes $K_B = b_1 P_A b_2$.

Since $a_1 b_1 = b_1 a_1$ and $a_2 b_2 = b_2 a_2$, we have $K = K_A = K_B$, the shared secret key.

4.2.3. Commutative subgroups. Instead of using *commuting* subgroups $A, B \leq G$, one can use *commutative* subgroups. Thus, suppose $A, B \leq G$ are two public commutative subgroups (or subsemigroups) of a group G , and let $w \in G$ be a public element.

- (1) Alice randomly selects private elements $a_1 \in A$, $b_1 \in B$. Then she sends the element $a_1 w b_1$ to Bob.
- (2) Bob randomly selects private elements $a_2 \in A$, $b_2 \in B$. Then he sends the element $a_2 w b_2$ to Alice.
- (3) Alice computes $K_A = a_1 a_2 w b_2 b_1$, and Bob computes $K_B = a_2 a_1 w b_1 b_2$. Since $a_1 a_2 = a_2 a_1$ and $b_1 b_2 = b_2 b_1$ in G , one has $K_A = K_B = K$ (as an element of G), which is now Alice's and Bob's common secret key.

4.2.4. Using matrices. Maze et al. [189] suggest to use the *semigroup* of matrices over a ring (or a semiring). This has an obvious advantage: matrices also have a ring structure, which means two binary operations rather than one can be used to “diffuse” transmissions. On the other hand, when matrices are used as the platform for a cryptographic primitive, there is always a danger of a linear algebra attack, as we will see in our Section 4.4. This is why the authors of [189] suggested to use matrices over a rather peculiar semiring that would not be embeddable into a field. Their semiring however was “too small”, which made their scheme vulnerable to a fairly straightforward, almost “brute force”, attack [264]. We believe, however, that the idea of using matrices in this context has potential, but the ground ring has to be chosen more carefully.

Here is the protocol from [189]. There is a public ring (or a semiring) R and public $n \times n$ matrices S , M_1 , and M_2 over R . The ring R should have a non-trivial center C . One way to guarantee that would be for R to be an algebra over a field K ; then, of course, $K \subseteq C$.

- (1) Alice chooses polynomials $p_A(x), q_A(x) \in C[x]$ and sends the matrix $U = p_A(M_1) S q_A(M_2)$ to Bob.
- (2) Bob chooses polynomials $p_B(x), q_B(x) \in C[x]$ and sends the matrix $V = p_B(M_1) S q_B(M_2)$ to Alice.
- (3) Alice computes $K_A = p_A(M_1) V q_A(M_2)$.
- (4) Bob computes $K_B = p_B(M_1) U q_B(M_2)$.

Since any two polynomials of the same matrix commute, we have $K = K_A = K_B$, the shared secret key.

4.2.5. Using the triple decomposition problem. This protocol is due to Kurt [164]. Her idea is to base security of a key exchange primitive on the “triple decomposition problem”, where a known element is to be factored into a product of three unknown factors. Recall that in the “usual” decomposition problem a known element is to be factored into a product of three factors, where two factors are unknown and one (in the middle) is known. The motivation for this modification is

the following: if the platform group which is used with the corresponding protocol is linear, then the “usual” decomposition problem can be reduced to a system of linear equations, thus making a linear algebra attack possible. This is illustrated, in particular, in our Section 4.4. With the triple decomposition problem, a similar trick can only reduce the problem to a system of *quadratic* equations which is, of course, less malleable at least for standard attacks.

In Kurt’s protocol the private key has three components. The idea is to hide each of these components by multiplying them by random elements of (public) subgroups. The important part is that one of the components is multiplied by random elements both on the right and on the left. Now we are getting to the protocol description.

There is a public platform group (or a monoid) G and two sets of subsets of G containing 5 subsets of G each, say $A = A_1, A_2, A_3, X_1, X_2$ and $B = B_1, B_2, B_3, Y_1, Y_2$, satisfying the following invertibility and commutativity conditions:

- (**Invertibility conditions**) The elements of X_1, X_2, Y_1, Y_2 are invertible.
- (**Commutativity conditions**) $[A_2, Y_1] = 1, [A_3, Y_2] = 1, [B_1, X_1] = 1$, and $[B_2, X_2] = 1$.

Alice and Bob agree on who will use which set of subsets; say Alice uses A and Bob uses B . Then the exchange between Alice and Bob goes as follows:

- (1) Alice chooses $a_1 \in A_1, a_2 \in A_2, a_3 \in A_3, x_1 \in X_1, x_2 \in X_2$, and computes:
 $u = a_1x_1, v = x_1^{-1}a_2x_2$, and $w = x_2^{-1}a_3$. Her private key is (a_1, a_2, a_3) .
- (2) Bob chooses $b_1 \in B_1, b_2 \in B_2, b_3 \in B_3, y_1 \in Y_1, y_2 \in Y_2$, and computes:
 $p = b_1y_1, q = y_1^{-1}b_2y_2$, and $r = y_2^{-1}b_3$. His private key is (b_1, b_2, b_3) .
- (3) Alice sends (u, v, w) to Bob.
- (4) Bob sends (p, q, r) to Alice.
- (5) Alice computes

$$a_1pa_2qa_3r = a_1(b_1y_1)a_2(y_1^{-1}b_2y_2)a_3(y_2^{-1}b_3) = a_1b_1a_2b_2a_3b_3 = K_A.$$

- (6) Bob computes

$$ub_1vb_2wb_3 = (a_1x_1)b_1(x_1^{-1}a_2x_2)b_2(x_2^{-1})a_3b_3 = a_1b_1a_2b_2a_3b_3 = K_B.$$

Thus, $K_A = K_B = K$ is Alice’s and Bob’s common secret key.

We refer to [164] for details on parameters of this scheme.

4.3. A protocol based on the factorization search problem

In this section, we describe a protocol based on the *factorization search problem*:

Given an element w of a group G and two subgroups $A, B \leq G$,
find any two elements $a \in A$ and $b \in B$ that would satisfy
 $a \cdot b = w$.

As before, there is a public group G , and two public subgroups $A, B \leq G$ commuting elementwise, i.e., $ab = ba$ for any $a \in A, b \in B$.

- (1) Alice randomly selects private elements $a_1 \in A, b_1 \in B$. Then she sends the element a_1b_1 to Bob.
- (2) Bob randomly selects private elements $a_2 \in A, b_2 \in B$. Then he sends the element a_2b_2 to Alice.
- (3) Alice computes

$$K_A = b_1(a_2b_2)a_1 = a_2b_1a_1b_2 = a_2a_1b_1b_2,$$

and Bob computes

$$K_B = a_2(a_1b_1)b_2 = a_2a_1b_1b_2.$$

Thus, $K_A = K_B = K$ is now Alice's and Bob's common secret key.

We note that the adversary, Eve, who knows the elements a_1b_1 and a_2b_2 , can compute $(a_1b_1)(a_2b_2) = a_1b_1a_2b_2 = a_1a_2b_1b_2$ and $(a_2b_2)(a_1b_1) = a_2a_1b_2b_1$, but neither of these products is equal to K if $a_1a_2 \neq a_2a_1$ and $b_1b_2 \neq b_2b_1$.

Finally, we point out a *decision* factorization problem:

Given an element w of a group G and two subgroups $A, B \leq G$, find out whether or not there are two elements $a \in A$ and $b \in B$ such that $w = a \cdot b$.

This seems to be a new and non-trivial problem in group theory, and therefore it gives an example of a group-theoretic problem motivated by cryptography, i.e., we have here remarkable feedback from cryptography to group theory.

4.4. Stickel's key exchange protocol

Stickel's protocol [265] is reminiscent of the classical Diffie-Hellman protocol (see our Section 1.2), although formally it is not a generalization of the latter. We show below, in Section 4.4.1, that Stickel's choice of platform (the group of invertible matrices over a finite field) makes the protocol vulnerable to linear algebra attacks. It appears however that even such a seemingly minor improvement as using non-invertible matrices instead of invertible ones would already make Stickel's protocol significantly less vulnerable, at least to linear algebra attacks, which is why we think this protocol has some potential. Our exposition in this section follows [249].

Let G be a public non-abelian finite group, $a, b \in G$ public elements such that $ab \neq ba$. The key exchange protocol goes as follows. Let N and M be the orders of a and b , respectively.

- (1) Alice picks two random natural numbers $n < N, m < M$ and sends $u = a^n b^m$ to Bob.
- (2) Bob picks two random natural numbers $r < N, s < M$ and sends $v = a^r b^s$ to Alice.
- (3) Alice computes $K_A = a^n v b^m = a^{n+r} b^{m+s}$.
- (4) Bob computes $K_B = a^r u b^s = a^{n+r} b^{m+s}$.

Thus, Alice and Bob end up with the same group element $K = K_A = K_B$ which can serve as the shared secret key.

When it comes to implementation details, exposition in [265] becomes somewhat foggy. In particular, it seems that the author actually prefers the following more general version of the above protocol.

Let $w \in G$ be public.

- (1) Alice picks two random natural numbers $n < N, m < M$, an element c_1 from the center of the group G , and sends $u = c_1 a^n w b^m$ to Bob.
- (2) Bob picks two random natural numbers $r < N, s < M$, an element c_2 from the center of the group G , and sends $v = c_2 a^r w b^s$ to Alice.
- (3) Alice computes $K_A = c_1 a^n v b^m = c_1 c_2 a^{n+r} w b^{m+s}$.
- (4) Bob computes $K_B = c_2 a^r u b^s = c_1 c_2 a^{n+r} w b^{m+s}$.

Thus, Alice and Bob end up with the same group element $K = K_A = K_B$.

We note that for this protocol to work, G does not have to be a group; a semigroup would do just as well (in fact, even better, as we argue below).

In [265], it was suggested that the group of invertible $k \times k$ matrices over a finite field F_{2^l} is used as the platform group G . We show in Section 4.4.1 that this choice of platform makes the protocol vulnerable to linear algebra attacks, but first we are going to discuss a general (i.e., not platform-specific) approach to attacking Stickel's protocol. We emphasize that this general approach works if G is any semigroup, whereas the attack in Section 4.4.1 is platform-specific; in particular, it only works if G is a group, but may not work for arbitrary semigroups.

Recall now that Alice transmits $u = c_1 a^n w b^m$ to Bob.

Our first observation is: to get a hold of the shared secret key K in the end, it is sufficient for the adversary (Eve) to find any elements $x, y \in G$ such that $xa = ax$, $yb = by$, $u = xwy$. Indeed, having found such x, y , Eve can use Bob's transmission $v = c_2 a^r w b^s$ to compute

$$xvy = xc_2 a^r w b^s y = c_2 a^r x w y b^s = c_2 a^r u b^s = K.$$

This implies, in particular, that multiplying by c_i does not enhance security of the protocol. More importantly, this also implies that it is not necessary for Eve to recover any of the exponents n, m, r, s ; instead, she can just solve a system of equations $xa = ax$, $yb = by$, $u = xwy$, where a, b, u, w are known and x, y unknown elements of the platform (semi)group G . This shows that, in fact, Stickel's protocol departs from the Diffie-Hellman protocol more than it seems. Moreover, solving the above system of equations in G is actually nothing else but solving the (subsemigroup-restricted) *decomposition search problem* (see our Section 2.3.3):

Given a recursively presented (semi)group G , two recursively generated sub(semi)groups $A, B \leq G$, and two elements $u, w \in G$, find two elements $x \in A$ and $y \in B$ that would satisfy $x \cdot w \cdot y = u$, provided at least one such pair of elements exists.

In reference to Stickel's scheme, the sub(semi)groups A and B are the *centralizers* of the elements a and b , respectively. The centralizer of an element $g \in G$ is the set of all elements $c \in G$ such that $gc = cg$. This set is a subsemigroup of G ; if G is a group, then this set is a subgroup.

So far, no particular (semi)group has been recognized as providing a secure platform for any of the protocols based on the decomposition search problem. It appears likely that semigroups of matrices over specific rings can generally make good platforms; see [248]. Stickel, too, used matrices in his paper [265], but he has made several poor choices, as we are about to see in the next Section 4.4.1. Also, Stickel's scheme is *at most* as secure as those schemes that are directly based on the alleged hardness of the decomposition search problem, because there are ways to attack Stickel's scheme without attacking the relevant decomposition search problem; for instance, Sramka [262] has offered an attack aimed at recovering one of the exponents n, m, r, s in Stickel's protocol. Our attack that we describe in Section 4.4.1 is more efficient, but on the other hand, it is aimed at recovering the shared secret key only, whereas Sramka's attack is aimed at recovering a private key.

4.4.1. Linear algebra attack. Now we are going to focus on the particular platform group G suggested by Stickel in [265]. In his paper, G is the group of invertible $k \times k$ matrices over a finite field F_{2^l} , where $k = 31$. The parameter l was

not specified in [265], but from what is written there, one can reasonably guess that $2 \leq l \leq k$. The choice of matrices a, b, w is not so important for our attack; what is important is that a and b are invertible. We note however that the choice of matrices a and b in [265] (more specifically, the fact that the entries of these matrices are either 0 or 1) provides an extra weakness to the scheme as we will see at the end of this section.

Recall that it is sufficient for Eve to find at least one solution of the system of equations $xa = ax$, $yb = by$, $u = xwy$, where a, b, u, w are known and x, y unknown $k \times k$ matrices over F_{2^l} . Each of the first two equations in this system translates into a system of k^2 linear equations for the (unknown) entries of the matrices x and y . However, the equation $u = xwy$ does not translate into a system of linear equations for the entries because it has a product of two unknown matrices. We therefore have to use the following trick: multiply both sides of the equation $u = xwy$ by x^{-1} on the left (here is where we use the fact that x is invertible!) to get

$$x^{-1}u = wy.$$

Now, since $xa = ax$ if and only if $x^{-1}a = ax^{-1}$, we denote $x_1 = x^{-1}$ and replace the system of equations mentioned in the previous paragraph by the following one:

$$x_1a = ax_1, \quad yb = by, \quad x_1u = wy.$$

Now each equation in this system translates into a system of k^2 linear equations for the (unknown) entries of the matrices x_1 and y . Thus, we have a total of $3n^2$ linear equations with $2k^2$ unknowns. Note however that a solution of the displayed system will yield the shared key K if and only if x_1 is invertible because $K = xvy$, where $x = x_1^{-1}$.

Since u is a known invertible matrix, we can multiply both sides of the equation $x_1u = wy$ by u^{-1} on the right to get $x_1 = wyu^{-1}$, and then eliminate x_1 from the system:

$$wyu^{-1}a = awyu^{-1}, \quad yb = by.$$

Now we have just one unknown matrix y , so we have $2k^2$ linear equations for k^2 entries of y . Thus, we have a heavily overdetermined system of linear equations (recall that in Stickel's paper, $k = 31$, so $k^2 = 961$). We know that this system must have at least one non-trivial (i.e., non-zero) solution; therefore, if we reduce the matrix of this system to an echelon form, there should be at least one free variable. On the other hand, since the system is heavily overdetermined, we can expect that the number of free variables will not be very big, so that it is feasible to go over possible values of free variables one at a time, until we find some values that yield an invertible matrix y . (Recall that entries of y are either 0 or 1; this is an extra weakness of Stickel's scheme that we mentioned before.) Note that checking the invertibility of a given matrix is easy because it is equivalent to reducing the matrix to an echelon form. In fact, in all our experiments there was just one free variable, so the last step (checking the invertibility) was not needed because if there is a unique non-zero solution of the above system, then the corresponding matrix y should be invertible.

Thus, the most obvious suggestion on improving Stickel's scheme is, as we mentioned before, to use non-invertible elements a, b, w ; this implies, in particular, that the platform should be a semigroup with (a lot of) non-invertible elements. If one is to use matrices, then it makes sense to use the semigroup of *all* $k \times k$

matrices over a finite ring (not necessarily a field!). Such a semigroup typically has a lot of non-invertible elements, so it should be easy to choose a, b, w non-invertible, in which case the linear algebra attack would not work. One more advantage of not restricting the pool to invertible matrices is that one can use not just powers a^j of a given public matrix in Stickel's protocol, but arbitrary expressions of the form $\sum_{i=1}^p c_i \cdot a^i$, where c_i are constants, i.e., elements of the ground ring. We note however that recently, Mullan [207] suggested a more sophisticated linear algebra attack on Stickel's scheme that appears to be successful also when non-invertible matrices are used.

Of course, there is no compelling reason why matrices should be employed in Stickel's scheme, but as we have explained above, with an abstract platform (semi)group G , Stickel's scheme is broken if the relevant decomposition search problem is solved, and so far, no particular abstract (semi)group has been recognized as resistant to known attacks on the decomposition search problem.

4.5. The Anshel-Anshel-Goldfeld protocol

In this section, we are going to describe a key establishment protocol that really stands out because, unlike other protocols in this chapter, it does not employ any commuting or commutative subgroups of a given platform group and can, in fact, use any non-abelian group with efficiently solvable word problem as the platform. This really makes a difference and gives a big advantage to the protocol of [5] over other protocols in this chapter.

A group G and elements $a_1, \dots, a_k, b_1, \dots, b_m \in G$ are public.

- (1) Alice picks a private $x \in G$ as a word in a_1, \dots, a_k (i.e., $x = x(a_1, \dots, a_k)$) and sends b_1^x, \dots, b_m^x to Bob.
- (2) Bob picks a private $y \in G$ as a word in b_1, \dots, b_m and sends a_1^y, \dots, a_k^y to Alice.
- (3) Alice computes $x(a_1^y, \dots, a_k^y) = x^y = y^{-1}xy$, and Bob computes $y(b_1^x, \dots, b_m^x) = y^x = x^{-1}yx$. Alice and Bob then come up with a common private key $K = x^{-1}y^{-1}xy$ (called the *commutator* of x and y) as follows: Alice multiplies $y^{-1}xy$ by x^{-1} on the left, while Bob multiplies $x^{-1}yx$ by y^{-1} on the left, and then takes the inverse of the whole thing: $(y^{-1}x^{-1}yx)^{-1} = x^{-1}y^{-1}xy$.

It may seem that solving the (simultaneous) conjugacy search problem for $b_1^x, \dots, b_m^x; a_1^y, \dots, a_k^y$ in the group G would allow an adversary to get the secret key K . However, if we look at Step (3) of the protocol, we see that the adversary would have to know either x or y not simply as a word in the generators of the group G , but as a word in a_1, \dots, a_k (respectively, as a word in b_1, \dots, b_m); otherwise, he would not be able to compose, say, x^y out of a_1^y, \dots, a_k^y . That means the adversary would also have to solve the *membership search problem*:

Given elements x, a_1, \dots, a_k of a group G , find an expression (if it exists) of x as a word in a_1, \dots, a_k .

We note that the membership *decision* problem is to determine whether or not a given $x \in G$ belongs to the subgroup of G generated by given a_1, \dots, a_k . This problem turns out to be quite hard in many groups. For instance, the membership decision problem in a braid group B_n is algorithmically unsolvable if $n \geq 6$ because such a braid group contains subgroups isomorphic to $F_2 \times F_2$ (that would be, for

example, the subgroup generated by $\sigma_1^2, \sigma_2^2, \sigma_4^2$, and σ_5^2 , see [42]), where F_2 is the free group of rank 2. In the group $F_2 \times F_2$, the membership decision problem is algorithmically unsolvable by an old result of Mihailova [198].

We also note that if the adversary finds, say, some $x' \in G$ such that $b_1^{x'} = b_1^{x'}, \dots, b_m^{x'} = b_m^{x'}$, there is no guarantee that $x' = x$ in G . Indeed, if $x' = c_b x$, where $c_b b_i = b_i c_b$ for all i (in which case we say that c_b centralizes b_i), then $b_i^{x'} = b_i^{x}$ for all i , and therefore $b^x = b^{x'}$ for any element b from the subgroup generated by b_1, \dots, b_m ; in particular, $y^x = y^{x'}$. Now the problem is that if x' (and, similarly, y') does not belong to the subgroup A generated by a_1, \dots, a_k (respectively, to the subgroup B generated by b_1, \dots, b_m), then the adversary may not obtain the correct common secret key K . On the other hand, if x' (and, similarly, y') does belong to the subgroup A (respectively, to the subgroup B), then the adversary will be able to get the correct K even though his x' and y' may be different from x and y , respectively. Indeed, if $x' = c_b x$, $y' = c_a y$, where c_b centralizes B and c_a centralizes A (elementwise), then

$$(x')^{-1}(y')^{-1}x'y' = (c_b x)^{-1}(c_a y)^{-1}c_b x c_a y = x^{-1}c_b^{-1}y^{-1}c_a^{-1}c_b x c_a y = x^{-1}y^{-1}xy = K$$

because c_b commutes with y and with c_a (note that c_a belongs to the subgroup B , which follows from the assumption $y' = c_a y \in B$, and, similarly, c_b belongs to A), and c_a commutes with x .

We emphasize that the adversary ends up with the correct key K (i.e., $K = (x')^{-1}(y')^{-1}x'y' = x^{-1}y^{-1}xy$) if and only if c_b commutes with c_a . The only visible way to ensure this is to have $x' \in A$ and $y' \in B$. Without verifying at least one of these inclusions, there seems to be no way for the adversary to make sure that he got the correct key.

Therefore, it appears that if the adversary chooses to solve the conjugacy search problem in the group G to recover x and y , he will then have to face either the membership search problem or the membership decision problem; the latter may very well be algorithmically unsolvable in a given group. The bottom line is that the adversary should actually be solving a more difficult version of the conjugacy search problem:

Given a group G , a subgroup $A \leq G$, and two elements $g, h \in G$,
find $x \in A$ such that $h = x^{-1}gx$, given that at least one such x
exists.

Finally, we note that what we have said in this section does not affect some heuristic attacks on the Anshel-Anshel-Goldfeld protocol suggested by several authors [86, 136] because these attacks, which use “neighborhood search” type (in a group-theoretic context also called “length based”) heuristic algorithms, are targeted, by design, at finding a solution of a given equation (or a system of equations) as a word in given elements. The point that we make in this section is that even if a fast (polynomial-time) *deterministic* algorithm is found for solving the conjugacy search problem in braid groups, this will not be sufficient to break the Anshel-Anshel-Goldfeld protocol by a *deterministic attack*.

4.6. Relations between different problems

In this section, we discuss relations between underlying problems in some of the protocols described earlier in this chapter.

We start with the *conjugacy search problem* (CSP), which was used in the protocol described in Section 4.1. A more accurate name for this problem would actually be the *subgroup-restricted* conjugacy search problem:

Given two elements w, h of a group G , a subgroup $A \leq G$, and the information that $w^a = h$ for some $a \in A$, find at least one particular element a like that.

In reference to the Ko-Lee protocol described in Section 4.1, one of the parties (Alice) transmits w^a for some private $a \in A$, and the other party (Bob) transmits w^b for some private $b \in B$, where the subgroups A and B commute elementwise, i.e., $ab = ba$ for any $a \in A, b \in B$.

Now suppose the adversary finds $a_1, a_2 \in A$ such that $a_1 w a_2 = a^{-1} w a$ and $b_1, b_2 \in B$ such that $b_1 w b_2 = b^{-1} w b$. Then the adversary gets

$$a_1 b_1 w b_2 a_2 = a_1 b^{-1} w b a_2 = b^{-1} a_1 w a_2 b = b^{-1} a^{-1} w a b = K,$$

the shared secret key.

We emphasize that these a_1, a_2 and b_1, b_2 do not have to do anything with the private elements originally selected by Alice or Bob, which simplifies the search substantially. We also point out that, in fact, it is sufficient for the adversary to find just one pair, say, $a_1, a_2 \in A$, to get the shared secret key:

$$a_1 (b^{-1} w b) a_2 = b^{-1} a_1 w a_2 b = b^{-1} a^{-1} w a b = K.$$

In summary, to get the secret key K , the adversary does not have to solve the (subgroup-restricted) conjugacy search problem, but instead, it is sufficient to solve an apparently easier (subgroup-restricted) decomposition search problem:

Given two elements w and w' of a group G , find any elements a_1 and a_2 that would belong to a given subgroup $A \leq G$ and satisfy $a_1 \cdot w \cdot a_2 = w'$, provided at least one such pair of elements exists.

We note that the protocol due to Shpilrain and Ushakov considered in Section 4.2.1 is based on a somewhat more general variant of the decomposition search problem where there are two different subgroups:

Given two elements w and w' of a group G and two subgroups $A, B \leq G$, find any two elements $a \in A$ and $b \in B$ that would satisfy $a \cdot w \cdot b = w'$, provided at least one such pair of elements exists.

Then, one more trick reduces the decomposition search problem to a special case where $w = 1$. Namely, given $w' = a \cdot w \cdot b$, multiply it on the left by the element w^{-1} (which is the inverse of the public element w) to get

$$w'' = w^{-1} a \cdot w \cdot b = (w^{-1} a \cdot w) \cdot b.$$

Thus, if we denote by A^w the subgroup conjugate to A by the element w , the problem for the adversary is now a *factorization search problem*:

Given an element w' of a group G and two subgroups $A^w, B \leq G$, find any two elements $a \in A^w$ and $b \in B$ that would satisfy $a \cdot b = w'$, provided at least one such pair of elements exists.

Since in the original Ko-Lee protocol one has $A = B$, this yields the following interesting observation: if in that protocol A is a normal subgroup of G , then $A^w = A$, and the above problem becomes: given $w' \in A$, find any two elements

$a_1, a_2 \in A$ such that $w' = a_1 a_2$. This problem is trivial: a_1 here could be any element from A , and then $a_2 = a_1^{-1} w'$.

Therefore, in choosing the platform group G and two commuting subgroups for a protocol described in our Section 4.1 or Section 4.2, one has to avoid normal subgroups. This means, in particular, that “artificially” introducing commuting subgroups as, say, direct factors is inappropriate from the security point of view.

At the other extreme, there are *malnormal* subgroups. A subgroup $A \leq G$ is called malnormal in G if, for any $g \in G$, $A^g \cap A = \{1\}$. We observe that if, in the original Ko-Lee protocol, A is a malnormal subgroup of G , then the decomposition search problem corresponding to that protocol has a unique solution if $w \notin A$. Indeed, suppose $w' = a_1 \cdot w \cdot a'_1 = a_2 \cdot w \cdot a'_2$, where $a_1 \neq a_2$, say. Then $a_2^{-1} a_1 w = w a'_2 a'_1^{-1}$, hence $w^{-1} a_2^{-1} a_1 w = a'_2 a'_1^{-1}$. Since A is malnormal, the element on the left does not belong to A , whereas the one on the right does, a contradiction. This argument shows that, in fact, already if $A^w \cap A = \{1\}$ for this particular w , then the corresponding decomposition search problem has a unique solution.

Finally, we describe one more trick that reduces, to some extent, the decomposition search problem to the (subgroup-restricted) conjugacy search problem. The same trick reduces the factorization search problem, too, to the subgroup-restricted conjugacy search problem. Suppose we are given $w' = awb$, and we need to recover $a \in A$ and $b \in B$, where A and B are two elementwise commuting subgroups of a group G .

Pick any $b_1 \in B$ and compute:

$$[awb, b_1] = b^{-1} w^{-1} a^{-1} b_1^{-1} awbb_1 = b^{-1} w^{-1} b_1^{-1} wbb_1 = (b_1^{-1})^{wb} b_1 = ((b_1^{-1})^w)^b b_1.$$

Since we know b_1 , we can multiply the result by b_1^{-1} on the right to get $w'' = ((b_1^{-1})^w)^b$. Now the problem becomes: recover $b \in B$ from the known $w'' = ((b_1^{-1})^w)^b$ and $(b_1^{-1})^w$. This is the subgroup-restricted conjugacy search problem. By solving it, one can recover a $b \in B$.

Similarly, to recover an $a \in A$, one picks any $a_1 \in A$ and computes:

$$\begin{aligned} [(awb)^{-1}, (a_1)^{-1}] &= awba_1 b^{-1} w^{-1} a^{-1} a_1^{-1} \\ &= awa_1 w^{-1} a^{-1} a_1^{-1} = (a_1)^{w^{-1} a^{-1}} a_1^{-1} = ((a_1)^{w^{-1}})^{a^{-1}} a_1^{-1}. \end{aligned}$$

Multiply the result by a_1 on the right to get $w'' = ((a_1)^{w^{-1}})^{a^{-1}}$, so that the problem becomes: recover $a \in A$ from the known $w'' = ((a_1)^{w^{-1}})^{a^{-1}}$ and $(a_1)^{w^{-1}}$.

We have to note that, since a solution of the subgroup-restricted conjugacy search problem is not always unique, solving the above two instances of this problem may not necessarily give the right solution of the original decomposition problem. However, any two solutions, call them b' and b'' , of the first conjugacy search problem differ by an element of the centralizer of $(b_1^{-1})^w$, and this centralizer is unlikely to have a non-trivial intersection with B .

A similar computation shows that the same trick reduces the factorization search problem, too, to the subgroup-restricted conjugacy search problem. Suppose we are given $w' = ab$, and we need to recover $a \in A$ and $b \in B$, where A and B are two elementwise commuting subgroups of a group G . Pick any $b_1 \in B$ and compute

$$[ab, b_1] = b^{-1} a^{-1} b_1^{-1} abb_1 = (b_1^{-1})^b b_1.$$

Since we know b_1 , we can multiply the result by b_1^{-1} on the right to get $w'' = (b_1^{-1})^b$. This is the subgroup-restricted conjugacy search problem. By solving it, one can recover a $b \in B$.

This same trick can, in fact, be used to attack the subgroup-restricted conjugacy search problem itself. Suppose we are given $w' = a^{-1}wa$, and we need to recover $a \in A$. Pick any b from the centralizer of A ; typically, there is a public subgroup B that commutes with A elementwise; then just pick any $b \in B$. Then compute

$$[w', b] = [a^{-1}wa, b] = a^{-1}w^{-1}ab^{-1}a^{-1}wab = a^{-1}w^{-1}b^{-1}wab = (b^{-w})^a b.$$

Multiply the result by b^{-1} on the right to get $(b^{-w})^a$, so the problem now is to recover $a \in A$ from $(b^{-w})^a$ and b^{-w} . This problem might be easier than the original problem because there is flexibility in choosing $b \in B$. In particular, a feasible attack might be to choose several different $b \in B$ and try to solve the above conjugacy search problem for each in parallel by using some general method (e.g., a length-based attack). Chances are that the attack will be successful for at least one of the b 's.

CHAPTER 5

Platform Groups

In Section 4.1, we have outlined some of the requirements on the platform group in a protocol based on the conjugacy search problem. Most of these requirements apply, in fact, to platform groups in any “canonical” (i.e., based on a one-way function) cryptographic protocol, so we summarize these general requirements here.

- (PG0) The group has to be well known (or well studied, or both).
- (PG1) The word problem in G should have a fast (linear- or quadratic-time) solution by a deterministic algorithm. Better yet, there should be an efficiently computable “normal form” for elements of G .
- (PG2) There should be a way to disguise elements of G so that it would be impossible to recover, say, x and y from a product xy just by inspection. Again, an efficiently computable normal form might be useful here.

In the absence of a normal form, say if G is just given by means of generators and relations without any additional information about properties of G , then at least some of these relations should be very short.

- (PG3) G should be a group of super-polynomial (i.e., exponential or “intermediate”) growth. This means that the number of elements of length n in G should grow faster than any polynomial in n ; this is needed to prevent attacks by complete exhaustion of the key space. Here “length n ” is typically just the length of a word representing a group element, but in a more general situation, this could be the length of some other description, i.e., “information complexity”.

In this chapter, we are going to examine several groups (or classes of groups) as possible platforms of cryptographic public-key protocols.

5.1. Braid groups

Braid groups have appeared as the platform for a “non-commutative” cryptographic public-key protocol in the seminal paper [5]. The legend has it that, after the authors of [5] had invented their protocol, they approached Joan Birman asking her to recommend “good” nonabelian groups that they could use as the platform. Not surprisingly, the answer was “braid groups”. After some initial excitement (which has even resulted in naming a new area of “braid group cryptography” — see [49], [57], [85]), it seems now that the conjugacy search problem in a braid group may not provide a sufficient level of security (see [192], [193]), unless keys are selected from some rather narrow subsets (yet to be determined) of the whole group. In what follows, we briefly discuss advantages and disadvantages of braid groups, and then give some group-theoretic background for an interested reader.

It is a fact that abstract groups, unlike numbers, are not something that most people learn at school. Therefore, there is an obvious communication problem

involved in marketing a cryptographic product that uses abstract groups one way or another. Braid groups clearly have an edge here because to explain what they are, one can draw simple pictures, thus alleviating the fear of the unknown. The fact that braid groups cut across many different areas of mathematics (and physics) helps, too, since this gives more credibility to the hardness of the relevant problem (e.g., the conjugacy search problem, see our Section 4.1).

We can recall that, for example, confidence in the security of the RSA cryptosystem is based on literally centuries-long history of attempts by thousands of people, including such authorities as Euler and Gauss, at factoring integers fast. The history of braid groups goes back to 1927, and again, thousands (well, maybe hundreds) of people, including prominent mathematicians like Artin, Birman, Thurston, V. F. R. Jones, and others have been working on various aspects, including algorithmic ones, of these groups.

On the other hand, from the security point of view, the fact that braid groups cut across so many different areas can be a disadvantage, because different areas provide different tools for solving a problem at hand. Furthermore, braid groups turned out to be linear [18], [163], which makes them potentially vulnerable to linear algebraic attacks (see e.g. [135], [144], [170]), and this alone is a serious security hazard.

As we have said before, braid groups appear in several areas of mathematics, and they admit many equivalent definitions. We start with an explicit presentation by generators and relators.

5.1.1. A group of braids and its presentation. In this section we follow the exposition of [69]. A braid is obtained by laying down a number of parallel pieces of string and intertwining them, without loosing track of the fact that they run essentially in the same direction. In our pictures the direction is horizontal. We number strands at each horizontal position from the top down. See Figure 5.1 for example.

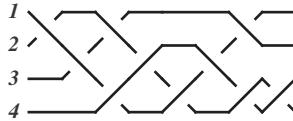


FIGURE 5.1. A 4-strand braid.

If we put down two braids u and v in a row so that the end of u matches the beginning of v we get another braid denoted by uv , i.e., concatenation of n -strand braids is a product. We consider two braids equivalent if there exists an isotopy between them, i.e., it is possible to move the strands of one of the braids in space (without moving the endpoints of strands and moving strands through each other) to get the other braid. We distinguish a special n -strand braid which contains no crossings and call it a trivial braid. Clearly the trivial braid behaves as left and right identity relative to the defined multiplication. The set B_n of isotopy classes of n -strand braids has a group structure, because if we concatenate a braid with its mirror image in a vertical plane the result is isotopic to the trivial braid.

Basically each braid is a sequence of strand crossings. A crossing is called positive if the front strand has a positive slope, otherwise it is called negative.

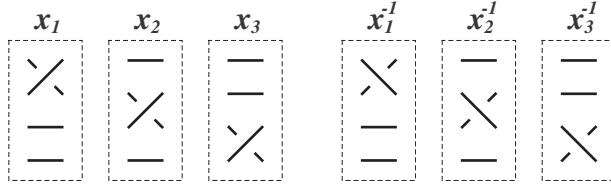
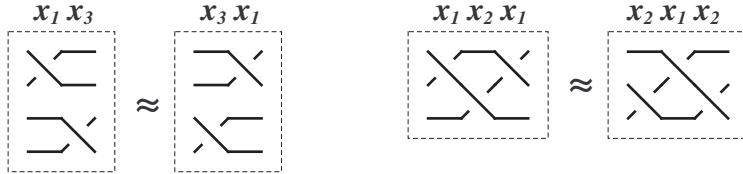
FIGURE 5.2. Generators of B_4 and their inverses.

FIGURE 5.3. Typical relations in braids.

There are exactly $n - 1$ crossing types for n -strand braids, we denote them by x_1, \dots, x_{n-1} , where x_i is a positive crossing of i th and $i + 1$ st strands. See Figure 5.2 for an example for B_4 . Since, as we mentioned above, any braid is a sequence of crossings the set $\{x_1, \dots, x_{n-1}\}$ generates B_n . It is easy to see that crossings x_1, \dots, x_{n-1} are subject to the relations

$$[x_i, x_j] = 1$$

for every i, j such that $|i - j| > 1$ and

$$x_i x_{i+1} x_i = x_{i+1} x_i x_{i+1}$$

for every i such that $1 \leq i \leq n - 2$. The corresponding braid configurations are shown in Figure 5.3. It is more difficult to prove that these two types of relations actually describe the equivalence on braids, i.e., the braid group B_n has the following (Artin) presentation:

$$B_n = \left\langle x_1, \dots, x_{n-1} \mid \begin{array}{ll} x_i x_j x_i = x_j x_i x_j & \text{if } |i - j| = 1 \\ x_i x_j = x_j x_i & \text{if } |i - j| > 1 \end{array} \right\rangle.$$

From this description, one easily sees that there are many pairs of commuting subgroups in B_n , which makes it possible to use B_n as the platform group for protocols in Sections 4.1 and 4.2. For example, Ko, Lee et al. [161] used the following two commuting subgroups: LB_n generated by $x_1, \dots, x_{[\frac{n}{2}]-1}$ and UB_n generated by $x_{[\frac{n}{2}]+1}, \dots, x_{n-1}$.

Now let us go over the properties (PG1)–(PG3) from the introduction to this chapter. There is a vast literature on the word and the conjugacy problems for braid groups. Early solutions of both problems, essentially due to Garside, can be found in J. Birman's monograph [21]. There were many other algorithms suggested since then, most of them quite efficient; the best currently known deterministic algorithm for the word problem has quadratic-time complexity with respect to the length of the input word [69]. We also single out Dehornoy's algorithm for the word problem

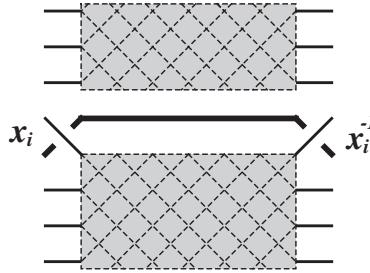


FIGURE 5.4. A handle.

[55] whose worst-case complexity is unknown at the time of this writing, but it appears to work very fast in practical implementations, which suggests that *generic-case complexity* of this algorithm might be linear-time. We note, in passing, that there are algorithms with linear-time generic-case complexity for the word problem in braid groups, which are based on quotient tests; see [146]. However, these algorithms by design give a fast answer only if the answer is “no”, i.e., if the input word is not equal to 1 in B_n . Dehornoy’s algorithm, on the other hand, gives a fast “yes” answer, too (at least, most of the time). However, no subexponential upper bound for the time complexity of Dehornoy’s algorithm has yet been established theoretically.

The conjugacy problem for braid groups was (and is) getting a lot of attention, too. Recently, a lot of research was done specifically addressing the (worst-case) complexity of this problem; remarkable progress has been made through the work of Birman, Gonzalez-Meneses, Gebhardt, E. Lee, S. J. Lee [22, 24, 23], [81], [92, 93], [169] and others. However, it is still an open problem whether the conjugacy decision and search problems in braid groups can be solved in polynomial time by a deterministic algorithm. A weaker problem whether the conjugacy problem in braid groups is in the complexity class **NP** is open, too (see [13, Problem (C3)]).

Getting to the property (PG2), we note that there are several normal forms for elements of braid groups known by now, and these are natural hiding mechanisms. A mild warning here is that the presence of different normal forms might be a potential security hazard, because one normal form may reveal what another one is trying to conceal. In fact, this observation was already used in [193] for cryptanalysis of one of the braid group based protocols: it was shown that converting Garside normal form to Dehornoy normal form makes the corresponding transmission(s) more vulnerable to some attacks.

Finally, regarding the property (PG3), we note that all braid groups B_n have exponential growth if $n \geq 2$. For $n \geq 3$ this immediately follows from the fact that B_n has free subgroups; for example, x_1^2 and x_2^2 generate a free subgroup; see [42]. Reasonable estimates of the growth rate of B_n are given in [279].

5.1.2. Dehornoy handle free form. Let w be a word in generators of B_n . An x_i -handle is a subword of w of the form

$$x_i^{-\varepsilon} w(x_1, \dots, x_{i-2}, x_{i+1}, \dots, x_n) x_i^\varepsilon$$

where $\varepsilon = \pm 1$. Schematically, an x_i -handle can be shown as in Figure 5.4. An x_i -handle $x_i^{-\varepsilon} w x_i^\varepsilon$ where $w = w(x_1, \dots, x_{i-2}, x_{i+1}, \dots, x_n)$ is called *permitted* if w

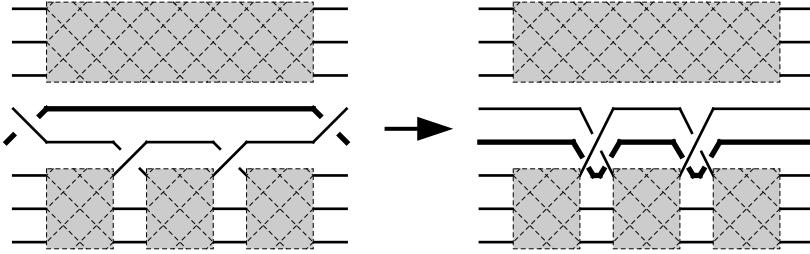


FIGURE 5.5. One step handle reduction of a permitted x_i -handle.

does not contain x_{i+1} -handles. We say that the braid word v' is obtained from the braid word v by a *one step handle reduction* if some subword of v is a permitted x_i -handle $x_i^{-\varepsilon}wx_i^\varepsilon$ and v' is obtained from v by applying the following substitutions for all letters in a handle $x_i^{-\varepsilon}wx_i^\varepsilon$:

$$x_j^{\pm 1} \rightarrow \begin{cases} 1 & \text{if } j = i, \\ x_{i+1}^\varepsilon x_i^{\pm 1} x_{i+1}^\varepsilon & \text{if } j = i+1, \\ x_j^{\pm 1} & \text{if } j < i \text{ or } j > i+1. \end{cases}$$

Schematically a reduction of an x_i -handle can be shown as in Figure 5.5. We say that the braid word v' is obtained from the braid word v by m *step handle reduction* if there exists a sequence of $m+1$ words $v = v_0, v_1, \dots, v_m = v'$ each of which is obtained from the previous one by a one step handle reduction. A braid word is called *handle free* if it contains no handles. The main statement about handle reduction can be formulated in the following theorem.

THEOREM 5.1.1. *Let v be a braid word. The following holds:*

- Any sequence of handle reductions applied to v eventually stops and produces a handle free braid word v' (which in general depends on a particular sequence of reductions) representing the same element of the braid group as v .
- The word v represents identity of a braid group if and only any sequence of handle reductions applied to v produces the trivial word.

REMARK 5.1.2 (Complexity estimates). Even though the handle reduction procedure in practice is very efficient and most of the time works in linear time in terms of the length of a braid word there is no good theoretical complexity estimate. For more on strategies for handle reduction and the related discussion on complexity issues see [59, Section 3.3].

5.1.3. Garside normal form. Consider a group of permutations S_n on n symbols. With each permutation $s \in S_n$ we can associate the shortest positive braid ξ_s such that $\pi(\xi_s) = s$. The elements

$$S = \{\xi_s \mid s \in S_n\} \subset B_n$$

are called *simple elements*. For permutations s and t we say that a simple element ξ_s is *smaller* than ξ_t (or, that ξ_s is a left divisor of ξ_t) and denote it by $\xi_s < \xi_t$ if there exists $r \in S_n$ such that $\xi_t = \xi_s \xi_r$.

There are two special simple braids in B_n : the trivial braid which is the smallest element of S , and the half-twist braid $\Delta = \xi_{(n,n-1,\dots,2,1)}$ which is the greatest

element of S . The set of simple braids with the order $<$ defined above has a lattice structure with gcd and lcm defined by

$$\gcd(\xi_s, \xi_t) = \max \{\xi_r \mid \xi_r < \xi_s \text{ and } \xi_r < \xi_t\}$$

and

$$\operatorname{lcm}(\xi_s, \xi_t) = \min \{\xi_r \mid \xi_s < \xi_r \text{ and } \xi_t < \xi_r\}.$$

Note that since S contains the minimal and the maximal elements it follows that gcd and lcm functions are well defined. Figure 5.6 shows the lattice of simple elements for B_4 .

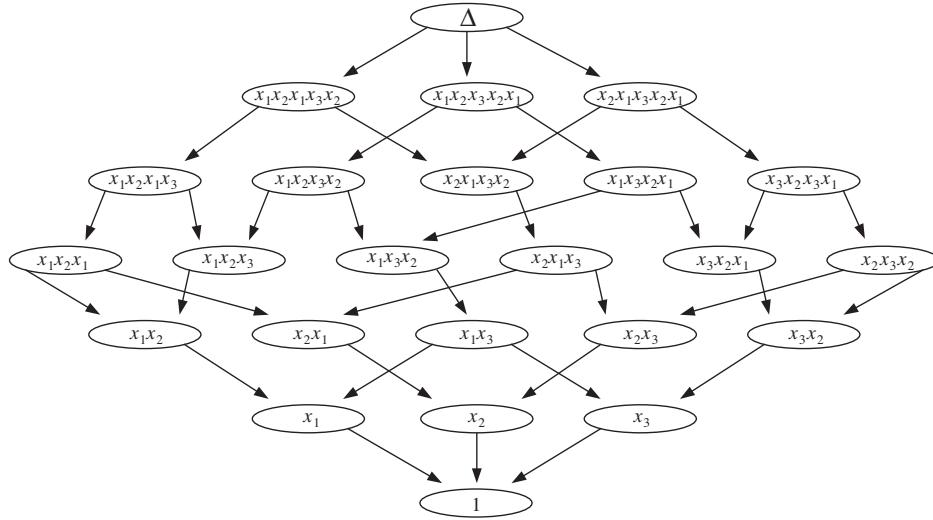


FIGURE 5.6. The lattice of simple elements in B_4 .

The *Garside left normal form* of a braid $a \in B_n$ is a pair $(p, (s_1, \dots, s_l))$ where $p \in \mathbb{Z}$ and s_1, \dots, s_l is a sequence of permutations in $S_n - \{1, \Delta\}$ satisfying the following property: for each $i = 1, \dots, l-1$,

$$\xi_1 = \gcd(\xi_{s_i^{-1}\Delta}, \xi_{s_{i+1}}).$$

A normal form $(p, (s_1, \dots, s_l))$ represents the following element in B_n :

$$\xi_\Delta^p \cdot \xi_{s_1} \cdot \dots \cdot \xi_{s_n}.$$

THEOREM 5.1.3 (Complexity estimate). *There exists an algorithm which for any braid word $w = w(x_1, \dots, x_{n-1})$ computes the normal form of the corresponding braid. Furthermore, the time complexity of the algorithm is $O(n^2|w|^2)$.*

5.2. Thompson's group

Thompson's group F , like the braid groups B_n , is well known in many areas of mathematics, including algebra, geometry, and analysis, so it does satisfy the property (PG0) from the introduction to this chapter. This group, too, is infinite non-abelian.

Now let us briefly go over the properties (PG1)–(PG3). First we note that Thompson's group has the following nice presentation in terms of generators and defining relations:

$$(3) \quad F = \langle x_0, x_1, x_2, \dots \mid x_i^{-1}x_kx_i = x_{k+1} \ (k > i) \rangle.$$

This presentation is infinite. There are also finite presentations of this group; for example,

$$F = \langle x_0, x_1, x_2, x_3, x_4 \mid x_i^{-1}x_kx_i = x_{k+1} \ (k > i, \ k < 4) \rangle,$$

but it is the infinite presentation above that allows for a convenient normal form, so we are going to use that presentation here.

For a survey on various properties of Thompson's group, we refer to [36]. Here we only give a description of the “classical” normal form for elements of F .

The classical normal form for an element of Thompson's group is a word of the form

$$(4) \quad x_{i_1} \dots x_{i_s} x_{j_t}^{-1} \dots x_{j_1}^{-1},$$

such that the following two conditions are satisfied:

(NF1) $i_1 \leq \dots \leq i_s$ and $j_1 \leq \dots \leq j_t$,

(NF2) if both x_i and x_i^{-1} occur, then either x_{i+1} or x_{i+1}^{-1} occurs, too.

We say that a word w is in *seminormal form* if it is of the form (4) and satisfies (NF1).

There is an easy procedure for reducing an arbitrary word w to the normal form in Thompson's group. First one reduces the given word to a seminormal form (which is not unique!). This is done by means of the following rewriting system (for all pairs (i, k) such that $i < k$):

$$\begin{aligned} x_k x_i &\rightarrow x_i x_{k+1}, \\ x_k^{-1} x_i &\rightarrow x_i x_{k+1}^{-1}, \\ x_i^{-1} x_k &\rightarrow x_{k+1} x_i^{-1}, \\ x_i^{-1} x_k^{-1} &\rightarrow x_{k+1}^{-1} x_i^{-1}, \end{aligned}$$

and, in addition, for all $i \in \mathbb{N}$,

$$x_i^{-1} x_i \rightarrow 1.$$

It is fairly obvious that this procedure terminates with a seminormal form for a given word w . After a seminormal form is obtained, one has to eliminate “bad pairs”, i.e., those pairs (x_i, x_i^{-1}) for which the property (NF2) above fails. For this, we need Lemma 5.2.1. For convenience we introduce a parametric function

$$\delta_\varepsilon : \{x_0, x_1, \dots\}^* \rightarrow \{x_0, x_1, \dots\}^*$$

(where $\varepsilon \in \mathbb{Z}$) defined by

$$x_{i_1} \dots x_{i_k} \xrightarrow{\delta_\varepsilon} x_{i_1+\varepsilon} \dots x_{i_k+\varepsilon}.$$

The function δ_ε may not be defined for some negative ε , but when it is used, it is assumed that the function is defined.

LEMMA 5.2.1 ([250]). *Let $w = x_{i_1} \dots x_{i_s} x_{j_t}^{-1} \dots x_{j_1}^{-1}$ be a seminormal form and let $(x_{i_a}, x_{j_b}^{-1})$ be the pair of generators in w which contradicts (NF2), where a and b are maximal with this property. Let*

$$w' = x_{i_1} \dots x_{i_{a-1}} \delta_{-1}(x_{i_{a+1}} \dots x_{i_s} x_{j_t}^{-1} \dots x_{j_{b+1}}^{-1}) x_{j_{b-1}}^{-1} \dots x_{j_1}^{-1}.$$

Then w' is in a seminormal form and $w = w'$ in F . Moreover, if $(x_{i_c}, x_{j_d}^{-1})$ is the pair of generators in w' which contradicts (NF2) (where a and b are maximal with this property), then $c < a$ and $d < b$.

Proof. It follows from the definition of (NF2) and seminormal forms that all indices in $x_{i_{a+1}} \dots x_{i_s} x_{j_t}^{-1} \dots x_{j_{b+1}}^{-1}$ are greater than $i_a + 1$ and, therefore, indices in $\delta_{-1}(x_{i_{a+1}} \dots x_{i_s} x_{j_t}^{-1} \dots x_{j_{b+1}}^{-1})$ are greater than i_a . Now it is clear that w' is a seminormal form. Then doing rewrites opposite to those from the rewriting system for obtaining a seminormal form (see above), we can get the word w' from the word w . Thus, $w = w'$ in the group F .

Now there are two possible cases: either $c > a$ and $d > b$ or $c < a$ and $d < b$. We need to show that the former case is, in fact, impossible. Assume, by way of contradiction, that $c > a$ and $d > b$. Note that if $(x_{i_a}, x_{j_b}^{-1})$ is a pair of generators in w contradicting (NF2), then $(x_{i_a+\varepsilon}, x_{j_b+\varepsilon}^{-1})$ contradicts (NF2) in $\delta_\varepsilon(w)$. Therefore, inequalities $c > a$ and $d > b$ contradict the choice of a and b . ■

By this lemma, we can start looking for bad pairs in a seminormal form starting in the middle of a word. The next algorithm implements this idea. It is in two parts; the first part finds all bad pairs starting in the middle of a given w , and the second part applies δ_ε to segments where it is required. A notable feature of Algorithm 5.2.2 is that it does not apply the operator δ_{-1} immediately (as in w' of Lemma 5.2.1) when a bad pair is found, but instead, it keeps the information about how indices must be changed later. This information is accumulated in two sequences (stacks), one for the positive subword of w , the other one for the negative subword of w . Also, in Algorithm 5.2.2, the size of stack S_1 (or S_2) equals the length of an auxiliary word w_1 (resp. w_2). Therefore, at step B), x_a (resp. x_b) is defined if and only if ε_1 (resp. ε_2) is defined.

ALGORITHM 5.2.2 (Erasing bad pairs from a seminormal form, [250]).

SIGNATURE. $w = \text{EraseBadPairs}(u)$.

INPUT. A seminormal form $u = x_{i_1} \dots x_{i_s} x_{j_t}^{-1} \dots x_{j_1}^{-1}$.

OUTPUT. A word w which is the normal form of u .

INITIALIZATION. Let $\delta = 0$, $\delta_1 = 0$, $\delta_2 = 0$, $w_1 = 1$, and $w_2 = 1$. Let $u_1 = x_{i_1} \dots x_{i_s}$ and $u_2 = x_{j_t}^{-1} \dots x_{j_1}^{-1}$ be the positive and negative parts of u . Additionally, we set up two empty stacks S_1 and S_2 .

COMPUTATIONS.

- A. Let the current $u_1 = x_{i_1} \dots x_{i_s}$ and $u_2 = x_{j_t}^{-1} \dots x_{j_1}^{-1}$.
- B. Let x_a be the leftmost letter of w_1 , x_b the rightmost letter of w_2 , and ε_i ($i = 1, 2$) the top element of S_i , i.e., the last element that was put there. If any of these values do not exist (because, say, S_i is empty), then the corresponding variable is not defined.
- 1) If $s > 0$ and ($t = 0$ or $i_s > j_t$), then:
 - a) multiply w_1 on the left by x_{i_s} (i.e. $w_1 \leftarrow x_{i_s} w_1$);
 - b) erase x_{i_s} from u_1 ;
 - c) push 0 into S_1 ;
 - d) goto 5).
- 2) If $t > 0$ and ($s = 0$ or $j_t > i_s$), then:
 - a) multiply w_2 on the right by $x_{j_t}^{-1}$ (i.e. $w_2 \leftarrow w_2 x_{j_t}^{-1}$);
 - b) erase $x_{j_t}^{-1}$ from u_2 ;
 - c) push 0 into S_2 ;

- d) goto 5).
- 3) If $i_s = j_t$ and (the numbers $a - \varepsilon_1$ and $b - \varepsilon_2$ (those that are defined) are not equal to i_s or $i_s + 1$), then:
 - a) erase x_{i_s} from u_1 ;
 - b) erase $x_{j_t}^{-1}$ from u_2 ;
 - c) if S_1 is not empty, increase the top element of S_1 ;
 - d) if S_2 is not empty, increase the top element of S_2 ;
 - e) goto 5).
- 4) If 1)–3) are not applicable (when $i_s = j_t$ and (one of the numbers $a - \varepsilon_1$, $b - \varepsilon_2$ is defined and is equal to either i_s or $i_s + 1$)), then:
 - a) multiply w_1 on the left by x_{i_s} (i.e. $w_1 \leftarrow x_{i_s} w_1$);
 - b) multiply w_2 on the right by $x_{j_t}^{-1}$ (i.e. $w_2 \leftarrow w_2 x_{j_t}^{-1}$);
 - c) erase x_{i_s} from u_1 ;
 - d) erase $x_{j_t}^{-1}$ from u_2 ;
 - e) push 0 into S_1 ;
 - f) push 0 into S_2 ;
 - g) goto 5).
- 5) If u_1 or u_2 is not empty then goto 1).
- C. While w_1 is not empty:
 - 1) let x_{i_1} be the first letter of w_1 (i.e. $w_1 = x_{i_1} \cdot w'_1$);
 - 2) take (pop) c from the top of S_1 and add to δ_1 (i.e. $\delta_1 \leftarrow \delta_1 + c$);
 - 3) multiply u_1 on the right by $x_{i_1 - \delta_1}$ (i.e. $u_1 \leftarrow u_1 x_{i_1 - \delta_1}$);
 - 4) erase x_{i_1} from w_1 .
- D. While w_2 is not empty:
 - 1) let $x_{j_1}^{-1}$ be the last letter of w_2 (i.e. $w_2 = w'_2 \cdot x_{j_1}^{-1}$);
 - 2) take (pop) c from the top of S_2 and add to δ_2 (i.e. $\delta_2 \leftarrow \delta_2 + c$);
 - 3) multiply u_2 on the left by $x_{j_1 - \delta_2}^{-1}$ (i.e. $u_2 \leftarrow x_{j_1 - \delta_2}^{-1} u_2$);
 - 4) erase $x_{j_1}^{-1}$ from w_2 .
- E. Return $u_1 u_2$.

We note that in [250], it was shown that the normal form of a given word w in Thompson's group F can be computed in time $O(|w| \log |w|)$.

Finally, we note that recently, Matucci [188] suggested an efficient attack on the decomposition (search) problem in Thompson's group F based on the interpretation of Thompson's group as the group of orientation-preserving piecewise-linear homeomorphisms of the unit interval, where the slopes are powers of two and the places where the slope changes are dyadic rationals. The lesson here, as well as with braid groups (see our Section 5.1), is that the presence of different (normal) forms for elements of a group might be a potential security hazard because one (normal) form may reveal what another one is trying to conceal.

5.3. Groups of matrices

In this section, we speculate that groups of matrices over finite commutative rings may be the best platforms for “canonical” cryptographic protocols described in our Chapter 4, because these groups have “the best of both worlds” in the sense that matrix multiplication is non-commutative, but matrix entries coming from a commutative ring provide a good hiding mechanism.

Recall that in Section 2.6, we emphasized the need for “diffusion”, i.e., for hiding factors in a product, and pointed out that in a group given by generators and relators the only visible mechanism for diffusion is using a normal form for elements of that group. With groups of matrices, we do not have this problem because these groups admit a different description, so that the way we are used to presenting their elements (as square tables) is, in fact, a “natural” normal form.

The reason why we want the ground ring to be finite is, again, that it is good for diffusion. Finite rings R are *periodic*, which means that for any $u \in R$, there are positive integers m, k such that $u^m = u^k$. Periodicity is good for diffusion because it gives rise to a *dynamical system*, and dynamical systems with a large number of states, even “innocent-looking” ones, usually exhibit very complex behavior; it is sufficient to mention the notorious “3x+1” problem [165].

We emphasize once again that

$$\text{COMMUTATIVITY} \quad \text{and} \quad \text{PERIODICITY}$$

are two major tools for hiding factors in a product; their importance for cryptographic security in general cannot be overestimated.

At the same time, for better security, commutativity might be reinforced by non-commutativity pretty much the same way as concrete is reinforced by steel to produce ferroconcrete. Thus,

$$\text{COMMUTATIVITY} \quad \text{in the corset of} \quad \text{NON-COMMUTATIVITY}$$

is another important ingredient of cryptographic security. It prevents the attacker from using obvious relations, such as $ab = ba$, to simplify a product.

Finally, we discuss specific finite commutative rings that can be used as ground rings for matrix groups in the cryptographic context. The simplest would be the ring \mathbb{Z}_n ; matrices over \mathbb{Z}_n can be used as the platform in the original Diffie-Hellman key exchange (see our Section 1.2), but one obvious disadvantage of this ring is that n has to be very large to provide for a sufficiently large key space. We note that there was very little research on how matrix groups over \mathbb{Z}_n compare to \mathbb{Z}_n itself in terms of security when used as the platform for the original Diffie-Hellman key exchange.

Another possibility would be to use $R = \mathbf{F}_p[x]/(f(x))$. Here \mathbf{F}_p is the field with p elements, $\mathbf{F}_p[x]$ is the ring of polynomials over \mathbf{F}_p , and $(f(x))$ is the ideal of $\mathbf{F}_p[x]$ generated by an irreducible polynomial $f(x)$ of degree n . This ring is actually isomorphic to the field \mathbf{F}_{p^n} , but the representation of R as a quotient field allows one to get a large key space while keeping all basic parameters rather small. We note that such a ring has been used by Tillich and Zémor [270] in their construction of a hash function (see also [248]). In the ring they used, $p = 2$, and n is a prime in the range $100 < n < 200$. Matrices that they consider are from the group $SL_2(R)$.

A further improvement of the same idea would be using the ring of *truncated polynomials* over \mathbb{Z}_n . Truncated polynomials, or, more precisely, N -truncated (one-variable) polynomials, are expressions of the form $\sum_{k=0}^N a_k x^k$, with the usual addition and multiplication by the rule $x^i \cdot x^j = x^{(i+j) \bmod (N+1)}$. Just like the ring R in the previous paragraph, this ring is a quotient of an algebra of polynomials by an ideal, but this ideal (generated by the polynomial x^{N+1}) is quite simple, and

computations in the corresponding quotient ring are quite efficient. Also, a large key space here is provided at a low cost; for example, with $n = 100$, $N = 20$, there are $100^{21} = 10^{42}$ N -truncated polynomials. This yields more than 10^{160} 2×2 matrices over this ring.

5.4. Small cancellation groups

Small cancellation groups were suggested as platforms in [255].

Small cancellation groups have relators satisfying a simple (and efficiently verifiable) “metric condition” (we follow the exposition in [177]). More specifically, let $F(X)$ be the free group with a basis $X = \{x_i \mid i \in I\}$, where I is an indexing set. Let $\varepsilon_k \in \{\pm 1\}$, where $1 \leq k \leq n$. A word $w(x_1, \dots, x_n) = x_{i_1}^{\varepsilon_1} x_{i_2}^{\varepsilon_2} \cdots x_{i_n}^{\varepsilon_n}$ in $F(X)$, with all x_{i_k} not necessarily distinct, is a *reduced X-word* if $x_{i_k}^{\varepsilon_k} \neq x_{i_{k+1}}^{-\varepsilon_{k+1}}$, for $1 \leq k \leq n-1$. In addition, the word $w(x_1, \dots, x_n)$ is *cyclically reduced* if it is a reduced X -word and $x_{i_1}^{\varepsilon_1} \neq x_{i_n}^{-\varepsilon_n}$. A set R containing cyclically reduced words from $F(X)$ is *symmetrized* if it is closed under cyclic permutations and taking inverses.

Let G be a group with presentation $\langle X; R \rangle$. A non-empty word $u \in F(X)$ is called a *piece* if there are two distinct relators $r_1, r_2 \in R$ of G such that $r_1 = uv_1$ and $r_2 = uv_2$ for some $v_1, v_2 \in F(X)$, with no cancelation between u and v_1 or between u and v_2 . The group G belongs to the class $C(p)$ if no element of R is a product of fewer than p pieces. Also, the group G belongs to the class $C'(\lambda)$ if for every $r \in R$ such that $r = uv$ and u is a piece, one has $|u| < \lambda|r|$.

5.4.1. Dehn's algorithm. If G belongs to the class $C'(\frac{1}{6})$, then Dehn's algorithm solves the word problem for G efficiently. This algorithm is very simple:

- (1) In an input (non-empty) word w , look for a “large” piece of a relator from R (that means, a piece whose length is more than a half of the length of the whole relator). If no such piece exists, then output “ $w \neq 1$ in G ”.
- (2) If such a piece, call it u , does exist, then $r = uv$ for some $r \in R$, where the length of v is smaller than that of u . Then replace u by v^{-1} in w . The length of the resulting word w' is smaller than that of w . If $w' = 1$, then output “ $w = 1$ in G ”.
- (3) If $w' \neq 1$, then repeat from Step 1 with $w := w'$.

Since the length of w decreases after each loop, this algorithm will terminate in a finite number of steps. It has quadratic time complexity with respect to the length of the input word w .

Finally, we note that a generic finitely presented group is a small cancellation group (see [8]).

5.5. Solvable groups

Recall that a group G is called *abelian* (or commutative) if $[a, b] = 1$ for any $a, b \in G$, where $[a, b]$ is the notation for $a^{-1}b^{-1}ab$. This can be generalized in different ways. A group G is called *metabelian* if $[[x, y], [z, t]] = 1$ for any $x, y, z, t \in G$. A group G is called *nilpotent of class c* if $[y_1, y_2, \dots, y_{c+1}] = 1$ for any $y_1, y_2, \dots, y_{c+1} \in G$, where $[y_1, y_2, y_3] = [[y_1, y_2], y_3]$, etc.

The commutator subgroup of G is the group $G' = [G, G]$ generated by all commutators, i.e., by expressions of the form $[u, v] = u^{-1}v^{-1}uv$, where $u, v \in G$. Furthermore, we can define, by induction, the k th term of the *lower central series* of G : $\gamma_1(G) = G$, $\gamma_2(G) = [G, G]$, $\gamma_k(G) = [\gamma_{k-1}(G), G]$. Note that one has $\alpha([u, v]) =$

$[\alpha(u), \alpha(v)]$ for any endomorphism α of G . Therefore, $\gamma_k(G)$ is a fully invariant subgroup of G for any $k \geq 1$, and so is $G''' = [G', G']$.

In this section, our focus is on *free metabelian groups*, because these groups were used as platforms in a cryptographic protocol in [254].

DEFINITION 5.5.1. Let F_n be the free group of rank n . The relatively free group F_n/F_n'' is called the *free metabelian group* of rank n , which we denote by M_n .

5.5.1. Normal forms in free metabelian groups. In this section, we describe a normal and a seminormal form for elements of a free metabelian group M_n . The seminormal form is good for transmissions, because it is easily convertible back to a word representing a transmitted element. However, this form is not unique if $n > 2$ (which is why we call it *seminormal*, so it cannot be used as a shared secret by Alice and Bob in a cryptographic protocol. For the latter purpose, the normal form (a 2×2 matrix) can be used).

Let $u \in M_n$. By u_{ab} we denote the abelianization of u , i.e., the image of u under the natural epimorphism $\alpha : M_n \rightarrow M_n/[M_n, M_n]$. Note that we can identify $M_n/[M_n, M_n]$ with $F_n/[F_n, F_n]$. Technically, u_{ab} is an element of a factor group of F_n , but we also use the same notation u_{ab} for any word in the generators x_i (i.e., an element of the ambient free group F_n) representing u_{ab} when there is no ambiguity.

For $u, v \in M_n$, by u^v we denote the expression $v^{-1}uv$; we also say that v acts on u by conjugation. If $u \in [M_n, M_n]$, then this action can be extended to the group ring $\mathbb{Z}(M_n/[M_n, M_n])$ which we are going to denote by $\mathbb{Z}A_n$, to simplify the notation. (Here $A_n = M_n/[M_n, M_n]$ is the free abelian group of rank n .) Let $W \in \mathbb{Z}A_n$ be expressed in the form $W = \sum a_i v_i$, where $a_i \in \mathbb{Z}$, $v_i \in A_n$. Then by u^W we denote the product $\prod (u^{a_i})^{v_i}$. This product is well-defined since any two elements of $[M_n, M_n]$ commute in M_n .

Now let $u \in M_n$. Then u can be written in the following seminormal form:

$$(5) \quad u = u_{ab} \cdot \prod_{i < j} [x_i, x_j]^{W_{ij}}$$

where $W_{ij} \in \mathbb{Z}A_n$. To get to this form, one can use a “collecting process” based on the following identities (recall that $[x, y] = x^{-1}y^{-1}xy$):

$$\begin{aligned} [y, x] &= [x, y]^{-1}, \\ xy &= yx[x, y], \\ xy^{-1} &= y^{-1}[y, x]^{y^{-1}x^{-1}}x, \\ x^{-1}y &= y[y, x]^{y^{-1}x^{-1}}x^{-1}, \\ [x, y]z &= z[x, y]^z. \end{aligned}$$

The collecting process itself is simple:

- (1) Using the above identities, go left to right along the word u collecting all “non-commutator” occurrences of x_1 on the left (that means, do not worry about occurrences of the form $[x_1, x_j]$ or $[x_j, x_1]$ created in the process). Repeat this with x_2, x_3 , etc. In the end of this process, u will be written in the form $u = u_{ab} \cdot c$, where $c \in [M_n, M_n]$ is a product of expressions of the form $[x_i, x_j]^g$, $g \in M_n$.

- (2) Since any two elements of $[M_n, M_n]$ commute in M_n , one can now easily re-group the expressions $[x_i, x_j]^g$ so that u takes the form $u = u_{ab} \cdot \prod_{i < j} [x_i, x_j]^{W_{ij}}$, where $W_{ij} \in \mathbb{Z}A_n$.

This process apparently takes quadratic time with respect to the length of u .

To convert the seminormal form (5) to a word is trivial because (5) is, in fact, already a word. The only problem with (5) is that it is not unique if $n > 2$, so it cannot be used as a shared secret by Alice and Bob. For the latter purpose, we are now going to introduce a normal form which is unique, efficiently computable (in quadratic time with respect to the length of u), but not so easily convertible back to a word.

We have to first introduce *Fox derivatives*, which are noncommutative analogs of usual Leibniz derivatives.

DEFINITION 5.5.2. Let $\mathbb{Z}F$ be the group ring of a free group F generated by x_1, x_2, \dots . A *Fox derivation* with respect to x_i is a map $\partial_{x_i} : \mathbb{Z}F \rightarrow \mathbb{Z}F$ such that $\partial_{x_i}(x_j) = \delta_{ij}$ and $\partial_{x_i}(vw) = \partial_{x_i}(v) + v \cdot \partial_{x_i}(w)$ for any $v, w \in F$. This map can be extended to the whole $\mathbb{Z}F$ by linearity.

EXAMPLE 5.5.3. Let $g \in F$ and let 1 be the identity of F . Since $\partial(1) = \partial(1) + \partial(1)$, it follows that $\partial(1) = 0$. Therefore $\partial(gg^{-1}) = \partial(g) + g\partial(g^{-1}) = 0$, which implies $\partial(g^{-1}) = -g^{-1}\partial(g)$.

EXAMPLE 5.5.4. Let x and y be generators of the free group $F(x, y)$. Then

$$\begin{aligned} \partial_x([x, y]) &= \partial_x(x^{-1}y^{-1}xy) \\ &= \partial_x(x^{-1}) + x^{-1}\partial_x(y^{-1}) + x^{-1}y^{-1}\partial_x(x) + x^{-1}y^{-1}x\partial_x(y) \\ &= -x^{-1} + x^{-1}y^{-1} = x^{-1}y^{-1}(1 - y), \\ \partial_y([x, y]) &= \partial_y(x^{-1}) + x^{-1}\partial_y(y^{-1}) + x^{-1}y^{-1}\partial_y(x) + x^{-1}y^{-1}x\partial_y(y) \\ &= -x^{-1}y^{-1} + x^{-1}y^{-1}x = -x^{-1}y^{-1}(1 - x). \end{aligned}$$

Let F_{ab} denote the abelianization of a free group F , i.e., the factor group F/F' . Let $\alpha : F \rightarrow F_{ab}$ be the natural epimorphism; it can be extended to the map $\alpha : \mathbb{Z}F \rightarrow \mathbb{Z}F_{ab}$ by linearity. A proof of the following proposition can be found in [122].

PROPOSITION 5.5.5. Let $w \in F_n$. Then $w \in F_n''$ if and only if $\alpha(\partial_{x_i}(w)) = 0$ for each generator x_i of F_n .

This proposition yields a simple algorithm for solving the word problem in a free metabelian group M_n : given $w \in M_n$ as a word in relatively free generators x_i , one considers w an element of the free group F_n with the same set of free generators, computes $\partial_{x_i}(w)$ for each x_i , and checks whether or not all of them abelianize to 0. The latter is straightforward since the word problem in the free abelian group F_{ab} is easily solvable.

This algorithm is not only simple but efficient, too:

PROPOSITION 5.5.6. The algorithm for solving the word problem in M_n based on Proposition 5.5.5 has at most quadratic time complexity with respect to the length of the input word.

Proof. Let $w \in F_n$ and let $|w| = m$ denote the usual lexicographic length of the word w . The computation of $\partial_{x_i}(w)$, for any generator x_i , produces at most

m summands in the free group ring $\mathbb{Z}F_n$. Thus, the computation of ∂_x has at most linear time complexity with respect to m . Then, deciding whether or not the abelianization of $\partial_{x_i}(w)$ is 0 amounts to collecting summands of the form $c \cdot h_i$, $c \in \mathbb{Z}$, $h_i \in F_n$, such that all h_i have the same abelianization. This is achieved by rewriting every h_i in the form $x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \cdot u_i$, where $u_i \in F'_n$. Since any h_i has length $\leq m$ and the number of h_i is at most m , this part of the algorithm takes time $O(m^2)$, which completes the proof. ■

Finally, we describe the normal form of $u \in M_n$ based on Fox derivatives.

For an element $u \in M_n$ of a free metabelian group, its normal form is a 2×2 matrix with the following entries:

- (1) The entry in the lower left corner is 0.
- (2) The entry in the lower right corner is 1.
- (3) The entry in the upper left corner is the abelianization of u , so it is an element of the free abelian group $M_n/[M_n, M_n]$.
- (4) The entry in the upper right corner is the most essential one. It is the vector of n abelianized partial Fox derivatives of the word u .

We note that the free abelian group $M_n/[M_n, M_n]$ acts on vectors of abelianized Fox derivatives by (componentwise) multiplication. This makes the set of normal forms a group under multiplication. Furthermore, the representation of elements of M_n by their normal forms is faithful, i.e., we actually have an embedding of M_n into a group of matrices; this is called *Magnus embedding* (see e.g. [122]).

5.6. Artin groups

Artin groups were used as platforms in a cryptographic protocol in [253].

Let $G(\Gamma)$ be a group with presentation

$$G(\Gamma) = \langle g_1, \dots, g_n ; r(g_i, g_j) = 1 \text{ (for } 1 \leq i, j \leq n \text{ and } i \neq j) \rangle,$$

where $n \geq 2$ and $r(g_i, g_j) = 1$ is a relator involving two generators. Given $G(\Gamma)$ there is an associated labeled graph Γ_G and vice versa. The vertices of the graph Γ_G are labeled by the generators of $G(\Gamma)$. Any two vertices $g_i, g_j \in \Gamma_G$ are connected by an edge if there is a relation $r(g_i, g_j) \in G$ between the corresponding generators; in other words, edges are labeled by relations.

EXAMPLE 5.6.1. An *Artin group* $A(\Gamma)$ is a group with presentation

$$A(\Gamma) = \langle a_1, \dots, a_n ; \mu_{ij} = \mu_{ji} \text{ for } 1 \leq i < j \leq n \rangle, \quad \text{where } \mu_{ij} = \underbrace{a_i a_j a_i \dots}_{m_{ij}}$$

and $m_{ij} = m_{ji}$. Artin groups arise as generalizations of braid groups, see e.g. [7]. For an Artin group $A(\Gamma)$, the associated labeled graph Γ_A has no multiple edges or loops. The vertices a_i of Γ_A are the generators of the Artin group. Any two vertices $a_i, a_j \in \Gamma_A$ are connected by an edge, labeled with the integer m_{ij} , associated to the relation $\mu_{ij} = \mu_{ji}$ (between the corresponding generators $a_i, a_j \in A(\Gamma)$).

In general, automorphisms (or endomorphisms) of the graph Γ_G induce automorphisms (or endomorphisms) of the group $G(\Gamma)$. Therefore, the graph associated to $G(\Gamma)$ gives us a way to construct a semigroup of endomorphisms of $G(\Gamma)$ that can contain a large pool of commuting elements. This was used in [253] as a basis of a key exchange protocol.

EXAMPLE 5.6.2. The relations of the braid groups B_n involve two generators. The corresponding graph associated to B_n is just a simple path, and it has only one automorphism that induces the following automorphism of B_n : $\sigma_i \mapsto \sigma_{n-i}$, which happens to be an inner automorphism of B_n . For other $G(\Gamma)$ groups, however, their corresponding graphs are more complex, and it is easy to arrange for a large semigroup (or a group) $T \subseteq \text{End } G(\Gamma)$ of endomorphisms (or automorphisms).

Artin groups $A(\Gamma)$ with the property that all the integers $m_{ij} \geq 4$ are called *Artin groups of extra large type*. A tree Γ_A can be associated to an Artin group of extra large type, providing a direct procedure for constructing a semigroup of endomorphisms of $A(\Gamma)$. Moreover, Artin groups of extra large type are automatic [222], thus the word problem for groups in this class can be solved in quadratic time, and by a result of [146], the word problem is solvable generically in linear time.

5.7. Grigorchuk's group

In this section we define the original Grigorchuk group Γ which first appeared in [108]. The group Γ is very important for many group theoretic problems, such as growth [109], amenability [110], just finite groups [111]. Recently it was also considered as a possible platform for a cryptographic scheme [223]. In our exposition we follow the book [129].

We start out by discussing the group of automorphisms of the infinite rooted binary tree. Denote by \mathcal{T} the infinite rooted binary tree. The set T of vertices of \mathcal{T} is the set of finite binary sequences as shown in Figure 5.7. Sets of vertices $L_k = \{b_1 \dots b_k \mid b_i = 0, 1\}$ of the same length k are called *levels* in the tree \mathcal{T} . An automorphism φ of \mathcal{T} fixes the root ε of \mathcal{T} and permutes the vertices preserving the connectedness, i.e., if vertices v_1 and v_2 are connected by an edge in \mathcal{T} then so $\varphi(v_1)$ and $\varphi(v_2)$ are. It is easy to see that for any k an automorphism φ of \mathcal{T} defines a permutation of the vertices of the level L_k . The group of automorphisms of \mathcal{T} is denoted by $\text{Aut}(\mathcal{T})$.

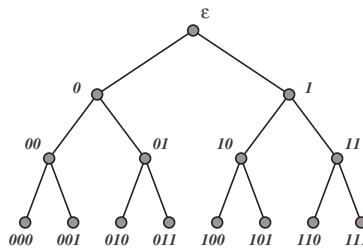


FIGURE 5.7. Binary tree with labeled vertices.

Consider an automorphism a which plays a special role in $\text{Aut}(\mathcal{T})$. It maps $\{0, 1\}$ -sequence to a $\{0, 1\}$ -sequence obtained by changing the first element:

$$a(b_1, b_2, \dots, b_n) = (1 - b_1, b_2, \dots, b_n).$$

Geometrically the action of a can be seen as swapping the left and the right subtrees of \mathcal{T} . Denote a subgroup of $\text{Aut}(\mathcal{T})$ stabilizing the vertices L_1 by $St(1)$. The subgroup $St(1)$ is normal of index 2 in $\text{Aut}(\mathcal{T})$ with left (and right) cosets $\{1, a\}$.

The Grigorchuk group Γ is a subgroup of $Aut(\mathcal{T})$ generated by four elements traditionally denoted by a, b, c, d . The element a is defined above. The other three automorphisms are defined as follows:

$$\begin{aligned}\mathbf{b}(b_1, b_2, \dots, b_n) &= \begin{cases} (b_1, 1 - b_2, \dots, b_n), & \text{if } b_1 = 0; \\ (b_1, \mathbf{c}(b_2, \dots, b_n)), & \text{if } b_1 = 1; \end{cases} \\ \mathbf{c}(b_1, b_2, \dots, b_n) &= \begin{cases} (b_1, 1 - b_2, \dots, b_n), & \text{if } b_1 = 0; \\ (b_1, \mathbf{d}(b_2, \dots, b_n)), & \text{if } b_1 = 1; \end{cases} \\ \mathbf{d}(b_1, b_2, \dots, b_n) &= \begin{cases} (b_1, b_2, \dots, b_n), & \text{if } b_1 = 0; \\ (b_1, \mathbf{b}(b_2, \dots, b_n)), & \text{if } b_1 = 1; \end{cases}\end{aligned}$$

It is straightforward to check that such defined maps belong to $Aut(\mathcal{T})$ and that $b, c, d \in St(1)$. The next example demonstrates the computation of the image of a vertex (binary sequence):

$$\begin{aligned}b(1, 1, 1, 0, 1) &= (1, c(1, 1, 0, 1)) = (1, 1, d(1, 0, 1)) \\ &= (1, 1, 1, b(0, 1)) = (1, 1, 1, 0, 0).\end{aligned}$$

LEMMA 5.7.1. *The following relations*

$$a^2 = b^2 = c^2 = d^2 = 1 \quad \text{and} \quad bc = cb = d$$

hold for generators of Γ .

Proof. Straightforward to check. ■

LEMMA 5.7.2. *The group $\Gamma = \langle a, b, c, d \rangle$ is a quotient of $(\mathbb{Z}/2\mathbb{Z}) * ((\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z}))$.*

Proof. It follows from Lemma 5.7.1 that a subgroup $\langle b, c, d \rangle$ is isomorphic to $(\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z})$. Furthermore, we have a relation $a^2 = 1$. Thus a mapping from $(\mathbb{Z}/2\mathbb{Z}) * ((\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z}))$ to Γ which maps the generator of the first free factor to a and abelian generators of the second free factor to b and c is an epimorphism. ■

It follows from Lemma 5.7.1 that any word $w = w(a, b, c, d)$ is equal to a word of the type

$$(6) \quad u_0 a u_1 a u_2 \dots u_{k-1} a u_k$$

where $u_0, \dots, u_k \in \{b, c, d\}$ and, perhaps, u_0, u_k are trivial. A word of the type (6) is called *reduced*.

LEMMA 5.7.3. *A word $w(a, b, c, d)$ represents an element of $St(1)$ if and only if the total number of a symbols is even. Furthermore, the subgroup $St(1)$ of Γ is generated by six elements $\{b, c, d, aba, aca, ada\}$.*

Proof. Follows from the definition of the elements a, b, c , and d . ■

It is straightforward to represent a word representing an element in $St(1)$ as a product of the generators of $St(1)$ of the form

$$(7) \quad u_0 \cdot (au_1a) \cdot u_2 \cdot (au_3a) \cdot u_4 \cdot \dots \cdot (au_{k-3}a) \cdot u_{k-2} \cdot (au_{k-1}a) \cdot u_k$$

where $u_0, \dots, u_k \in \{b, c, d\}$ and u_0, u_k are, perhaps, trivial. Since any element φ of $St(1)$ acts trivially on the first level of the tree \mathcal{T} it follows that φ can be

represented as a pair (φ_0, φ_1) where φ_0 and φ_1 are automorphisms of the left and the right subtrees of \mathcal{T} respectively defined by φ . It is easy to check that

$$(8) \quad \begin{aligned} b &= (a, c), & aba &= (c, a), \\ c &= (a, d), & aca &= (d, a), \\ d &= (1, b), & ada &= (b, 1). \end{aligned}$$

Moreover, if $u, v \in St(1)$ and $u = (u_0, u_1)$ and $v = (v_0, v_1)$ then

$$(9) \quad uv = (u_0v_0, u_1v_1).$$

LEMMA 5.7.4. *Let $w = w(a, b, c, d) \in St(1)$ be a reduced word and $w = (w_0, w_1)$. Then*

$$|w_0|, |w_1| \leq \frac{|w| + 1}{2}.$$

Moreover, if w starts with a symbol a then $|w_0|, |w_1| \leq \frac{|w|}{2}$.

Proof. Any word $w \in St(1)$ can be seen as a product (7). Using formulae (8) and (9) it is trivial to check both inequalities. ■

ALGORITHM 5.7.5 (Word problem for Γ).

INPUT. A reduced word $w = w(a, b, c, d)$.

OUTPUT. *True* if w represents the identity of Γ . *False* otherwise.

COMPUTATIONS.

- A. If $|w| = 0$ then output *True*. If $|w| = 1$ then output *False*.
- B. Cyclically permute w so it starts with the a symbol.
- C. Compute the algebraic sum of powers of a in w . If it is odd then output *False*.
- D. Rewrite w as a product of elements b, c, d, aba, aca, ada and using (8) find corresponding (w_0, w_1) .
- E. Recursively check if w_0 and w_1 represent identity of Γ and if so output *True*. Otherwise output *False*.

THEOREM 5.7.6. *Algorithm 5.7.5 solves the word problem for the Grigorchuk group Γ in time $O(|w| \log |w|)$.*

Proof. A word w represents the identity in Γ if and only if it belongs to $St(1)$ and acts trivially on the left and the right subtree of \mathcal{T} . Hence, the algorithm for solving the identity problem must start with checking if $w \in St(1)$ and if so rewrite w as a product of elements b, c, d, aba, aca, ada . Then using the equalities (8) compute elements (w_0, w_1) describing the action on subtrees and check if they represent identities. Thus Algorithm 5.7.5 correctly solves the Identity problem for Γ .

For a word w Algorithm 5.7.5 performs steps A–D in linear time $O(|w|)$. At step E we come up to two words w_0 and w_1 each of length less than $\frac{|w|}{2}$ and recursively run the same procedure for w_0 and w_1 .

The total time complexity of steps A–D for words w_0 and w_1 is $O(|w_0|) + O(|w_1|) = O(|w|)$ and on step D we split w_0 and w_1 and obtain four new words. The time complexity of steps A–D for those words is bounded by $O(|w|)$ again, and so on until we get words of lengths 0 and 1 for which we know the answer. Since each time we split words into two pieces each of which is not longer than half of the original word, it follows that in $\log_2 |w|$ steps we come up to words of lengths 0

or 1. Therefore the total complexity of the process is bounded by $O(|w| \log_2 |w|)$. \blacksquare

Based on Algorithm 5.7.5 one can arrange a procedure for constructing normal forms of elements of Γ . The normal form ρ_w of an element $w \in \Gamma$ is a finite binary tree with vertices labeled with elements $\{1, a, b, c, d\} \in \Gamma$. The tree ρ_w for a word $w = w(a, b, c, d)$ can be constructed recursively as follows:

- Using Algorithm 5.7.5 check if w is equal to $1, a, b, c, d$ in Γ and if so ρ_w contains one vertex labeled with the corresponding symbol.
- If w is not equal to $1, a, b, c, d$ in Γ then:
 - If $w \in St(1)$ and $w = (w_0, w_1)$ then ρ_w is a tree with a root labeled by 1, the left subtree is ρ_{w_0} , and the right subtree is ρ_{w_1} .
 - If $w \notin St(1)$ and $wa = (w_0, w_1)$ then ρ_w is a tree with a root labeled by a , the left subtree is ρ_{w_0} , and the right subtree is ρ_{w_1} .

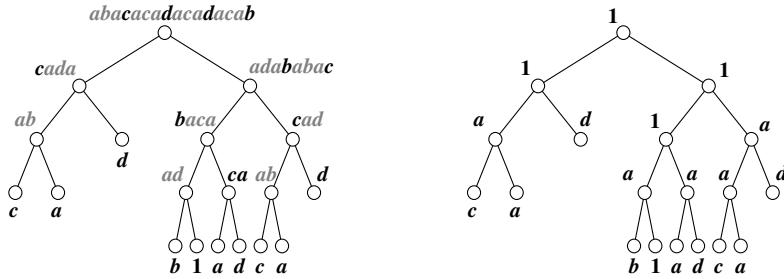


FIGURE 5.8. A normal form of the element $abacacacadaacab$ (on the right). The tree on the left visualizes the process of computing the normal forms.

We omit proofs for the next two statements.

PROPOSITION 5.7.7. *Let $u = u(a, b, c, d)$ and $v = v(a, b, c, d)$. The words u and v represent the same element of Γ if and only if $\rho_u = \rho_v$.*

THEOREM 5.7.8. *There exists an algorithm which for any word $w = w(a, b, c, d)$ constructs the corresponding normal form ρ_w in time $O(|w| \log_2 |w|)$.*

CHAPTER 6

More Protocols

In this chapter, we describe a couple of “non-commutative” cryptographic protocols that have so far attracted less attention than those described in Chapter 4. Also, the groups that are used as platforms in this chapter are rather different from those used in Chapter 4. However, the ideas behind the two protocols in this chapter are not brand new; the idea behind the protocol in Section 6.1 can be traced to [202], where a similar approach was used in a commutative situation, whereas the protocol in Section 6.2 is a generalization of the well-known ElGamal protocol [68] (see also our Section 1.3).

6.1. Using the subgroup membership search problem

Our exposition in this section follows [254].

We describe here a cryptosystem whose security is based on the computational hardness of the *subgroup membership (search) problem* (cf. our Section 2.3.4):

Given a recursively presented group G , a subgroup H generated by h_1, \dots, h_k , and an element $h \in H$, find an expression of h in terms of h_1, \dots, h_k .

First we outline the ideas behind our cryptosystem. These ideas can be traced to [202], where a similar approach was used in a commutative situation, but the resulting cryptosystem was not accepted by the cryptographic community as secure [105]. We believe that employing a nonabelian group instead of a polynomial algebra as the platform does make a difference in both the efficiency and security components. We note that various nonabelian groups have been used as platforms in various protocols (see our Chapter 4); in particular, braid groups [5], [161], Grigorchuk groups [91], Thompson’s group [250], polycyclic groups [65]. In this section, we use groups of a different nature.

To explain our approach, we need some more background. Let F_n denote the free group of rank n . It is well known that any map on the generators of F_n into F_n extends to an endomorphism of F_n (cf. our Section 2.1). It is also well known that free groups are not the only groups with this property.

DEFINITION 6.1.1. Let F be a free group and let R be a normal subgroup of F . The factor group F/R is called *relatively free* if R is fully invariant, i.e., if $\alpha(R) \leq R$ for any endomorphism α of F . If x_1, \dots, x_n are free generators of F , then x_1R, \dots, x_nR are called relatively free generators of F/R .

We are going to denote relatively free generators of F/R simply by x_1, \dots, x_n when there is no ambiguity. Let \mathcal{F}_n denote a relatively free group of rank n , i.e., $\mathcal{F}_n = F_n/R$ for some fully invariant R . Then any map on its generators into \mathcal{F}_n can be extended to an endomorphism of \mathcal{F}_n . Because of this property, any relatively free group \mathcal{F}_n can be a candidate for the platform of our cryptosystem.

We also need to recall one basic fact about automorphisms of F_n . Let $X = \{x_1, \dots, x_n\}$ be a set of generators of F_n and consider the maps $\alpha_j, \beta_{jk} : X \rightarrow F_n$ given by

$$\alpha_j : x_i \mapsto \begin{cases} x_i^{-1} & \text{if } i = j, \\ x_i & \text{if } i \neq j, \end{cases} \quad \text{and} \quad \beta_{jk} : x_i \mapsto \begin{cases} x_i x_j & \text{if } i = k, \\ x_i & \text{if } i \neq k, \end{cases}$$

where $1 \leq i, j, k \leq n$. Then all the α_j 's and β_{jk} 's define automorphisms of F_n , called *Nielsen automorphisms*, and generate the whole automorphism group of F_n .

Nielsen automorphisms can be defined for any relatively free group \mathcal{F}_n the same way. They generate a subgroup of the group $\text{Aut}(\mathcal{F}_n)$ of all automorphisms of \mathcal{F}_n ; this subgroup is called the group of *tame* automorphisms. In some cases, it is equal to the whole $\text{Aut}(\mathcal{F}_n)$ (see e.g. [12]); in other cases, it is a proper subgroup of $\text{Aut}(\mathcal{F}_n)$ (see e.g. [35, 246]). For our purposes, it is important that groups of the form $F_n/[R, R]$ (where R is a normal subgroup of F_n , and $[R, R]$ its commutator subgroup) tend to have many non-tame automorphisms by a result of [246]:

THEOREM 6.1.2 ([246]). *Let $\mathcal{F}_n = F_n/[R, R]$, and let $u \in R$. If $R \leq \gamma_3(F_n) = [[F_n, F_n], F_n]$ and $n \geq 2$, then the following automorphism $\alpha_{u,j}$ of \mathcal{F}_n is not tame:*

$$\alpha_{u,j} : x_j \rightarrow x_j[x_j, u, x_j], \quad x_i \rightarrow x_i, \quad i \neq j.$$

Note that $\alpha_{u,j}^{-1} : x_j \rightarrow x_j[x_j, u^{-1}, x_j], \quad x_i \rightarrow x_i, \quad i \neq j$.

Now we introduce more notation. Let $\langle x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}; R \rangle$ be a presentation of a relatively free group \mathcal{F}_{n+m} . Let $\varphi \in \text{Aut } \mathcal{F}_{n+m}$ be an automorphism such that

$$\varphi : x_i \mapsto y_i, \quad \text{where} \quad y_i = y_i(x_1, \dots, x_{n+m}).$$

This φ acts on \mathcal{F}_{n+m}^{n+m} , the direct product of $n+m$ copies of \mathcal{F}_{n+m} , in a natural way:

$$\varphi : (x_1, \dots, x_{n+m}) \mapsto (y_1, \dots, y_{n+m}),$$

and, more generally,

$$\varphi : (u_1, \dots, u_{n+m}) \mapsto (y_1(u_1, \dots, u_{n+m}), \dots, y_{n+m}(u_1, \dots, u_{n+m}))$$

for any $u_i = u_i(x_1, \dots, x_{n+m}) \in \mathcal{F}_{n+m}$.

Then, let $\hat{\varphi}$ be the restriction of φ to the subgroup of \mathcal{F}_{n+m}^{n+m} that consists of all elements of the form $(u_1(x_1, \dots, x_n), \dots, u_n(x_1, \dots, x_n), 1, \dots, 1)$ (this subgroup is isomorphic to \mathcal{F}_n^n), so that

$$\hat{\varphi} : (u_1, \dots, u_n, 1, \dots, 1) \mapsto (\hat{y}_1(u_1, \dots, u_n), \dots, \hat{y}_{n+m}(u_1, \dots, u_n)),$$

where $\hat{y}_i(x_1, \dots, x_n) = y_i(x_1, \dots, x_n, 1, \dots, 1)$.

We note that $\hat{\varphi}$ is a one-to-one map because the restriction of a one-to-one map to a subgroup is itself one-to-one. This property will be important to us later. At the same time, there is no visible way of recovering φ from $\hat{\varphi}$.

The protocol.

Let \mathcal{F}_{n+m} be a relatively free group. We discuss a specific choice of the platform group in Section 6.1.1; here we present our public-key encryption/decryption protocol without specifying the platform group.

Key Generation: (1) Alice privately chooses an automorphism $\varphi \in \text{Aut } \mathcal{F}_{n+m}$ as a random product of Nielsen automorphisms and some easily invertible automorphisms of the type given in the theorem in

the Introduction: $\varphi = \tau_1 \cdots \tau_k$ and computes the inverse $\varphi^{-1} = \tau_k^{-1} \cdots \tau_1^{-1}$. This φ^{-1} is Alice's private decryption key.

- (2) Let $y_i = y_i(x_1, \dots, x_{n+m}) = \varphi(x_i)$. Alice publishes $\hat{y}_i(x_1, \dots, x_n) = y_i(x_1, \dots, x_n, 1, \dots, 1)$, $i = 1, \dots, n+m$. This collection of \hat{y}_i is the public encryption key.

Encryption and Decryption: The information considered for encryption (i.e., Bob's private message) is an element w of the subgroup of \mathcal{F}_{n+m}^{n+m} that consists of all elements of the form $(u_1(x_1, \dots, x_n), \dots, u_n(x_1, \dots, x_n), 1, \dots, 1)$. Thus,

$$w = (w_1(x_1, \dots, x_n), \dots, w_n(x_1, \dots, x_n))$$

(one can suppress the 1's in the end).

- (1) Encryption is defined by

$$w \mapsto \hat{\varphi}(w) = (\hat{y}_1(w_1, \dots, w_n), \dots, \hat{y}_{n+m}(w_1, \dots, w_n)).$$

Bob then transmits $\hat{\varphi}(w)$ as a tuple of words in the alphabet X .

- (2) Decryption is defined by $\hat{\varphi}(w) \mapsto [\varphi^{-1}(\hat{\varphi}(w))]|_{x_{n+1}=\dots=x_{n+m}=1} = w$.

We note that the encryption of w here is *not* a homomorphic image (or a preimage) of w , in contrast to homomorphic public-key cryptosystems of [113].

The adversary can recover the plaintext w if he finds an expression for each x_1, \dots, x_n in terms of $\hat{y}_1, \dots, \hat{y}_{n+m}$, i.e., if he solves the membership search problem for the subgroup generated by $\hat{y}_1, \dots, \hat{y}_{n+m}$ (this subgroup is actually \mathcal{F}_n). Then from

$$x_i = u_i(\hat{y}_1(x_1, \dots, x_n), \dots, \hat{y}_n(x_1, \dots, x_n))$$

he can get

$$w_i = u_i(\hat{y}_1(w_1, \dots, w_n), \dots, \hat{y}_n(w_1, \dots, w_n)).$$

If however the adversary wants to completely break the cryptosystem, i.e., to get the private decryption key, then he has to find the automorphism φ (and its inverse) based on the restriction $\hat{\varphi}$. This problem looks unapproachable to us by any deterministic method other than trying out products of "elementary" automorphisms of \mathcal{F}_{n+m} until the one with the right restriction is found. This method is computationally infeasible for large n, m .

Now we give a "toy example" to illustrate how our protocol works. This example is not platform-specific because we only use Nielsen automorphisms as building blocks here.

EXAMPLE 6.1.3. Let $n = 2, m = 1$, and let β_{jk} be Nielsen automorphisms of \mathcal{F}_3 defined above. Let $\varphi = \beta_{23}^2 \beta_{12} \beta_{31} \beta_{32}$. Then $\varphi : x_1 \rightarrow x_1 x_3 x_2^2$, $x_2 \rightarrow x_2 x_1 x_3 x_2^2$, $x_3 \rightarrow x_3 x_2^2$. Thus, $y_1 = x_1 x_3 x_2^2$, $y_2 = x_2 x_1 x_3 x_2^2$, $y_3 = x_3 x_2^2$, and therefore $\hat{y}_1 = x_1 x_2^2$, $\hat{y}_2 = x_2 x_1 x_2^2$, $\hat{y}_3 = x_2^2$.

Thus, Bob's private message $(u_1(x_1, x_2), u_2(x_1, x_2))$ is encrypted as $(u_1 u_2^2, u_2 u_1 u_2^2, u_2^2) = (v_1, v_2, v_3)$.

In this simple example the adversary can easily obtain x_1, x_2 from the public \hat{y}_i as follows: $x_1 = \hat{y}_1 \hat{y}_3^{-1}$, $x_2 = \hat{y}_2 \hat{y}_1^{-1}$. Therefore, he can recover $u_1 = v_1 v_3^{-1}$, $u_2 = v_2 v_1^{-1}$.

As a comment to this example, we note that in a free group, the subgroup membership search problem can be efficiently solved by Nielsen's method (see e.g.

[177]). However, adding automorphisms $\alpha_{u,j}$ described in Theorem 6.1.2 makes a difference because it makes working in a free group less helpful.

6.1.1. Groups of the form $F/[R, R]$. In the original paper [254], relatively free groups called *free metabelian groups* were suggested as platforms for the cryptosystem described in the previous section. Recall that a group G is called *abelian* (or commutative) if $[a, b] = 1$ for any $a, b \in G$, where $[a, b]$ is the notation for $a^{-1}b^{-1}ab$. This can be generalized in different ways. A group G is called *metabelian* if $[[x, y], [z, t]] = 1$ for any $x, y, z, t \in G$. A group G is called *nilpotent of class $c \geq 1$* if $[y_1, y_2, \dots, y_{c+1}] = 1$ for any $y_1, y_2, \dots, y_{c+1} \in G$, where $[y_1, y_2, y_3] = [[y_1, y_2], y_3]$, etc.

The commutator subgroup of G is the group $G' = [G, G]$ generated by all commutators, i.e., by expressions of the form $[u, v] = u^{-1}v^{-1}uv$, where $u, v \in G$. Furthermore, we can define, by induction, the k th term of the *lower central series* of G : $\gamma_1(G) = G$, $\gamma_2(G) = [G, G]$, $\gamma_k(G) = [\gamma_{k-1}G, G]$. Note that one has $\alpha([u, v]) = [\alpha(u), \alpha(v)]$ for any endomorphism α of G . Therefore, $\gamma_k(G)$ is a fully invariant subgroup of G for any $k \geq 1$, and so is $G'' = [G', G']$.

DEFINITION 6.1.4. Let F_n be the free group of rank n . The relatively free group F_n/F_n'' is called the *free metabelian group* of rank n , which we denote by M_n .

The choice of free metabelian groups for the platform is motivated by the following useful facts:

- (1) The word problem in the group M_n is solvable in time $O(m^2n)$ with respect to the length m of a given word. Moreover, every element of M_n has a unique associated “normal form”, which makes it possible to use our protocol for an efficient encryption/decryption of actual messages without prior common key establishment.
- (2) M_n has exponential growth which provides for an exponential (with respect to the key size) key space.
- (3) Subgroup membership (search) problem in M_n has no known polynomial-time solution.

Here (2) is a well-established fact [201], so we leave it without further comments. As for (3), there are two different solutions of the membership (decision) problem in M_n known by now [45, 237], but none of them is “even close” to yielding a polynomial-time algorithm for solving the membership search problem.

However, if the free metabelian group M_n is used as the platform for the above scheme, Hofheinz and Unruh [134] have offered a successful heuristic attack that bypasses solving the subgroup membership problem in M_n . Their attack “pretends” that public key and ciphertext are transmitted as elements of a free group instead of the considered free metabelian group. Thus, further research is needed with respect to the choice of the platform group for this scheme.

6.2. The MOR cryptosystem

The MOR cryptosystem [217] is a natural generalization of the ElGamal cryptosystem (see [68] or our Section 1.3) to non-abelian groups.

Let G be a group and $\varphi : G \rightarrow G$ an automorphism. Alice’s keys are as follows:

Private Key: $m \in \mathbb{N}$.

Public Key: φ and φ^m .

Encryption: To send a message $a \in G$, Bob computes φ^r and φ^{mr} for a random $r \in \mathbb{N}$. The ciphertext is the pair $(\varphi^r, \varphi^{mr}(a))$.

Decryption: Alice knows m , so if she receives the ciphertext $(\varphi^r, \varphi^{mr}(a))$, she computes φ^{mr} from φ^r , then φ^{-mr} and then from $\varphi^{mr}(a)$ recovers $a = \varphi^{-mr}(\varphi^{mr}(a))$.

We note that Alice can compute φ^{-mr} two ways. If she knows the (finite) order t of the automorphism φ , then she can use the identity $\varphi^{-1} = \varphi^{t-1}$. Alternatively, Alice may select φ as a product of “elementary” Nielsen automorphisms (cf. our Section 6.1), each of which is easy to invert. In any case, since the choice of φ is up to Alice, she can choose whatever is easier for her, given a particular platform group G .

There is no particularly satisfying choice of a platform group for the MOR cryptosystem at this time. Matrix groups over finite fields are currently being explored as candidates for such platforms; see e.g. [181] and references therein.

CHAPTER 7

Using Decision Problems in Public-Key Cryptography

In this chapter, we suggest to use *decision problems* from combinatorial group theory as the core of a public key establishment protocol or a public key cryptosystem. Decision problems are problems of the following nature: given a property \mathcal{P} and an object \mathcal{O} , find out whether or not the object \mathcal{O} has the property \mathcal{P} . Decision problems may allow us to design cryptographic primitives secure against some of the possible “brute force” attacks (e.g. encryption emulation attack) by an adversary with essentially unlimited computational capabilities.

A particular decision problem that we consider here is the *word problem* which is: given a recursive presentation of a group G and a word g in generators of G , find out whether or not $g = 1$ in G . From the very description of the word problem we see that it consists of two parts: “whether” and “not”. We call them the “yes” and “no” parts of the word problem, respectively. If a group is given by a recursive presentation in terms of generators and relators, then the “yes” part of the word problem has a recursive solution because one can recursively enumerate all products of defining relators, their inverses and conjugates. However, the number of factors in such a product required to represent a word of length n which is equal to 1 in G , can be very large compared to n ; in particular, there are groups G with efficiently solvable word problem and words w of length n equal to 1 in G , such that the number of factors in any factorization of w into a product of defining relators, their inverses and conjugates is not bounded by any tower of exponents in n ; see [223]. Furthermore, if in a group G the word problem is recursively unsolvable, then the length of a proof verifying that $w = 1$ in G is not bounded by any recursive function of the length of w .

We also note that the “no” part of the word problem in many groups is recursively unsolvable, and therefore the “brute force” attack described above will not be effective against this part, at least in theory. At the same time, for “most” groups G a random word w is not going to be equal to 1 in G with overwhelming probability (we explain this in Section 7.1.3), which makes sampling (i.e., randomly selecting) words not equal to 1 in G quite efficient.

Finally, we point out that there is no recursively presented group (or semigroup) that would have both “yes” and “no” parts of the word problem non-recursive.

7.1. The Shpilrain-Zapata scheme

In this section, we describe a cryptographic protocol (see [255]) that employs computational hardness of the word problem in groups. In this protocol, Bob transmits to Alice an encrypted binary sequence which Alice decrypts correctly with probability “very close” to 1.

We note that long ago, there was an attempt to use the word problem in public-key cryptography [180], but it did not meet with success, for several reasons. One of the reasons, which is relevant to the discussion above, was pointed out recently in [13]: the problem that is actually used in [180] is not the word problem, but the *word choice problem*: given words g, w_1, w_2 , find out whether $g = w_1$ or $g = w_2$ in G , provided one of the two equalities holds. In this problem, both parts are recursively solvable for any recursively presented platform group G because they both are the “yes” parts of the word problem, and therefore the word choice problem cannot be used for our purposes. Thus, a similarity of the proposal of [255] to that of [180] is misleading, and the former seems to be the first proposal actually based on a decision problem.

7.1.1. The protocol.

Here is a sketch of the protocol from [255].

- (1) A pool of group presentations with efficiently solvable word problem is considered public (e.g. is part of Alice’s software).
- (2) Alice chooses randomly a particular presentation Γ from the pool, diffuses it by isomorphism-preserving (Tietze) transformations to obtain a diffused presentation Γ' , discards some of the relators and publishes the abridged diffused presentation $\hat{\Gamma}$.
- (3) Bob transmits his private binary sequence to Alice by transmitting an element equal to 1 in $\hat{\Gamma}$ (and therefore also in Γ') in place of “1” and an element not equal to 1 in Γ' in place of “0”.
- (4) Alice recovers Bob’s binary sequence by first converting elements of Γ' to the corresponding (under the isomorphism that she knows) elements of Γ , and then solving the word problem in Γ .

Most parts of this protocol are rather nontrivial and open several interesting research avenues. We refer to the original paper [255] for details.

A priori it looks like the most nontrivial part is finding an element which is not equal to 1 in Γ' since Bob does not even know the whole presentation Γ' . We solve this problem by “going with the flow”, so to speak. More specifically, we just let Bob select a random (well, almost random) word of sufficiently big length and show that, with overwhelming probability, such an element is not equal to 1 in Γ' . We discuss this in more detail in Section 7.1.3.

We emphasize once again what is, in our opinion, the main novelty of this protocol compared to the existing ones. The point is to deprive the adversary (Eve) from attacking the protocol by doing an exhaustive search of the sender’s private key space, which is the most obvious (although often “computationally infeasible”) way to attack all existing public-key protocols.

The way we plan to achieve our goal is relevant to part (3) of the above protocol, more specifically, to solving the word problem in $\hat{\Gamma}$. If Bob transmits an element g equal to 1 in $\hat{\Gamma}$, Eve may be able to detect this by going over all products of all conjugates of relators from $\hat{\Gamma}$ and their inverses. This set is recursive, but as we have pointed out in the Introduction, there are groups G with efficiently solvable word problem and words w of length n equal to 1 in G , such that the length of a proof verifying that $w = 1$ in G is not bounded by any tower of exponents in n ; see [223].

Furthermore, if Bob transmits an element g not equal to 1 in $\hat{\Gamma}$, then detecting this is even more difficult for Eve. In fact, it is impossible in general; Eve’s best

hope here is that she will be lucky to find a factor group of $\hat{\Gamma}$ where the word problem is solvable, and that $g \neq 1$ in that factor group. This is what we call a *quotient attack*.

Now let us take a closer look at Eve’s “encryption emulation” attack, which is:

Eve performs key generations over and over again, each time with fresh randomness, until the transmission to be attacked is obtained – this will happen eventually with overwhelming probability. The correctness of the scheme guarantees that the corresponding secret key (as obtained by Eve while performing key generation) allows her to decrypt illegitimately.

This would indeed be viable if the correctness of the legitimate decryption by Alice was perfect. However, in our situation this kind of attack may not work for a general $\hat{\Gamma}$. Suppose Eve is building up two lists, corresponding to two possible encryptions “ $0 \rightarrow w \neq 1$ in $\hat{\Gamma}$ ” or “ $1 \rightarrow w = 1$ in $\hat{\Gamma}$ ” by Bob. Our first observation is that the list that corresponds to “ $0 \rightarrow w \neq 1$ ” is useless to Eve because it is simply going to contain all words in the alphabet $X = \{x_1, \dots, x_n, x_1^{-1}, \dots, x_n^{-1}\}$ since Bob is choosing such w simply as a random word. Therefore, Eve may just as well forget about this list and concentrate on the other one, that corresponds to “ $1 \rightarrow w = 1$ ”.

Now the situation boils down to the following: if a word w transmitted by Bob appears on the list, then it is equal to 1 in G . If not, then not. The only problem is: how can Eve possibly conclude that w does not appear on the list if the list is infinite? One could say here that Eve can stop at some point and conclude that $w \neq 1$ with overwhelming probability, just like Alice does. The point however is that this probability may not at all be as “overwhelming” as the probability of the correct decryption by Alice. Compare:

- (1) For Alice to decrypt correctly “with overwhelming probability”, the probability $P_1(N)$ for a random word w of length N not to be equal to 1 should converge to 1 (reasonably fast) as N goes to infinity.
- (2) For Eve to decrypt correctly “with overwhelming probability”, the probability $P_2(N, f(N))$ for a random word w of length N , which is equal to 1, to have a proof of length $\leq f(N)$ verifying that $w = 1$, should converge to 1 (reasonably fast) as N goes to infinity. Here $f(N)$ represents Eve’s computational capabilities; this function can be arbitrary, but fixed.

We see that the functions $P_1(N)$ and $P_2(N, f(N))$ are of very different natures, and any correlation between them is unlikely. We note that the function $P_1(N)$ is generally well understood, and in particular, it is known that in any infinite group G , $P_1(N)$ indeed converges to 1 as N goes to infinity; see our Section 7.1.3 for more details.

On the other hand, the functions $P_2(N, f(N))$ are more complex; they are currently the subject of very active research, and in particular, it may happen that for any $f(N)$, there are groups in which $P_2(N, f(N))$ does not converge to 1 at all. Of course, $P_2(N, f(N))$ may depend on a particular algorithm used by Bob to produce words equal to 1, but we leave this discussion out of this book.

To conclude this section, we point out that encryption (of one bit) in this protocol is rather efficient; it takes quadratic time in the length of a transmitted word. Also, it is straightforward to see that the time Alice needs to decrypt each transmitted word w is bounded by $C \cdot |w|$, where $|w|$ is the length of w and C is

a constant which basically depends on Alice’s private isomorphism between Γ and Γ' .

The fact that Alice (the receiver) and the adversary are separated in power is essentially due to Alice’s knowledge of her private isomorphism between Γ and Γ' (note that Bob does *not* have to know this isomorphism for encryption!).

We have to admit here one disadvantage of this protocol compared to most well-established public-key protocols: we have encryption with a rather big “expansion factor”. Computer experiments show that, with parameters suggested in [255], one bit in Bob’s message gets encrypted into a word of length approximately 150 on average. This is the price we have to pay for “granting” the adversary too much computational power.

Finally, we touch upon semantic security in the end of Section 7.1.3.

7.1.2. Pool of group presentations. There are many classes of finitely presented groups with solvable word problem known by now, e.g. one-relator groups, hyperbolic groups, nilpotent groups, metabelian groups. Note however that Alice should be able to sample (i.e., to randomly select) a presentation from the pool *efficiently*, which imposes some restrictions on classes of presentations that can be used in this context. The class of finitely presented groups that we suggest to include in our pool is the class of *small cancellation groups*; see Section 5.4 of this book.

We briefly recall some basic definitions here. Small cancellation groups have relators satisfying a simple (and efficiently verifiable) “metric condition”. More specifically, let $F(X)$ be the free group with a basis $X = \{x_i \mid i \in I\}$, where I is an indexing set. Let $\varepsilon_k \in \{\pm 1\}$, where $1 \leq k \leq n$. A word $w(x_1, \dots, x_n) = x_{i_1}^{\varepsilon_1} x_{i_2}^{\varepsilon_2} \cdots x_{i_n}^{\varepsilon_n}$ in $F(X)$, with all x_{i_k} not necessarily distinct, is a *reduced X-word* if $x_{i_k}^{\varepsilon_k} \neq x_{i_{k+1}}^{-\varepsilon_{k+1}}$, for $1 \leq k \leq n-1$. In addition, the word $w(x_1, \dots, x_n)$ is *cyclically reduced* if it is a reduced X -word and $x_{i_1}^{\varepsilon_1} \neq x_{i_n}^{-\varepsilon_n}$. A set R containing cyclically reduced words from $F(X)$ is *symmetrized* if it is closed under cyclic permutations and taking inverses.

Let G be a group with presentation $\langle X; R \rangle$. A non-empty word $u \in F(X)$ is called a *piece* if there are two distinct relators $r_1, r_2 \in R$ of G such that $r_1 = uv_1$ and $r_2 = uv_2$ for some $v_1, v_2 \in F(X)$, with no cancellation between u and v_1 or between u and v_2 . The group G belongs to the class $C(p)$ if no element of R is a product of fewer than p pieces. Also, the group G belongs to the class $C'(\lambda)$ if for every $r \in R$ such that $r = uv$ and u is a piece, one has $|u| < \lambda|r|$.

In particular, if G belongs to the class $C'(\frac{1}{6})$, then Dehn’s algorithm solves the word problem for G efficiently (see Section 5.4). This algorithm has quadratic time complexity with respect to the length of the input word w .

We also note that a generic finitely presented group is a small cancellation group (see [8]); therefore, to randomly select a small cancellation group, Alice can just take a few random words and check whether the corresponding symmetrized set satisfies the condition for $C'(\frac{1}{6})$. If not, then repeat.

To conclude this section, we give a more specific recipe, with sample parameters, for Alice to produce a presentation Γ for the protocol in Section 7.1.1.

- (1) Alice fixes a number k , $10 \leq k \leq 20$, of generators in her presentation Γ . Her Γ will therefore have generators x_1, \dots, x_k .

- (2) Alice selects m random words r_1, \dots, r_m in the generators x_1, \dots, x_k . Here $10 \leq m \leq 30$ and the lengths l_i of r_i are random integers from the interval $L_1 \leq l_i \leq L_2$. Particular values that we suggest are: $L_1 = 12$, $L_2 = 20$.
- (3) After Alice obtains the abridged presentation $\hat{\Gamma}$, she adds a relation

$$x'_i = \prod_{j=1}^M [x'_i, w_j]$$

to it, where x'_i is a (randomly chosen) generator from $\hat{\Gamma}$, w_j are random elements of length 1 or 2 in the generators x'_1, x'_2, \dots , and $M = 10$. (Our commutator notation is: $[a, b] = a^{-1}b^{-1}ab$.) This relation is needed to foil *quotient attacks*. Then Alice finds the preimage of this relation under the isomorphism between Γ and $\hat{\Gamma}$ and adds this preimage to the defining relators of Γ . Thus, Γ finally has k generators and $m+1$ defining relators.

- (4) Finally, Alice checks whether her private presentation Γ satisfies the small cancellation condition $C'(\frac{1}{6})$ (it will with overwhelming probability, see [8]). If not, then she has to start over.

7.1.3. Generating random elements in finitely presented groups. In this section, we explain how to implement the crucial step (3) of the protocol given in Section 7.1.1.

When Bob wants to transmit an element equal to 1 in $\hat{\Gamma}$, he should construct a word, looking “as random as possible” (for semantic security), in the relators $\hat{r}_1, \dots, \hat{r}_l$ and their conjugates. When he wants to transmit an element not equal to 1 in $\hat{\Gamma}$, he just selects a random word of sufficiently big length; it turns out that, with overwhelming probability, such an element is not equal to 1 in Γ (we explain it in the end of this section).

We start with a description of Bob’s possible diffusion strategy for producing elements equal to 1 in $\hat{\Gamma}$. (It is rather straightforward to produce a random word of a given length in generators x_1, \dots, x_k , so we are not going to discuss it here.) When Bob transmits a word w equal to 1 in $\hat{\Gamma}$, he wants to diffuse it so that large pieces of defining relators would not be visible in w . In some specific groups (e.g. in braid groups) diffusion is provided by a “normal form” (see Section 2.6), which is a collection of symbols that uniquely corresponds to a given element of the group. The existence of such normal forms is usually due to some special algebraic or geometric properties of a given group.

However, since Bob does not know any meaningful properties of the group defined by the presentation $\hat{\Gamma}$ which is given to him, he cannot employ normal forms in the usual sense. The only useful property that the presentation $\hat{\Gamma}$ has is that most of its defining relators have length 3 or 4; see Section 7.1.2. We are going to take advantage of this property as follows. We suggest the following procedure, which is probably best described by the word “shuffling”.

- (1) Make a product of the form $u = s_1 \cdots s_p$, where each s_i is randomly chosen among defining relators $\hat{r}_1, \hat{r}_2, \dots$, of length 3 or 4, their inverses, and their conjugates by one- or two-letter words in x'_1, x'_2, \dots . The number p of factors should be sufficiently big, at least 10 times the number of defining relators in $\hat{\Gamma}$.

- (2) Insert approximately $\frac{2p}{k}$ expressions of the form $x'_j(x'_j)^{-1}$ or $(x'_j)^{-1}x'_j$ in random places of the word u (here k is the number of generators x'_i of $\hat{\Gamma}$), for random values of j .
- (3) Going left to right in the word u , look for two-letter subwords that are parts of defining relators \hat{r}_i of length 3 or 4. When you spot such a subword, replace it by the inverse of the augmenting part of the same defining relator and continue. For example, suppose there is a relator $\hat{r}_i = x'_1x'_2x'_3x'_4$, and suppose you spot the subword $x'_1x'_2$ in u . Then replace it by $(x'_4)^{-1}(x'_3)^{-1}$ (obviously, $x'_1x'_2 = (x'_4)^{-1}(x'_3)^{-1}$ in your group). If you spot the subword $x'_2x'_3$ in w , replace it by $(x'_1)^{-1}(x'_4)^{-1}$. If there is more than one choice for replacement, choose randomly between them.
- (4) Cancel remaining subwords (if any) of the form $x'_j(x'_j)^{-1}$ or $(x'_j)^{-1}x'_j$.

Steps (2)–(4) should be repeated approximately p times for good mixing.

Finally, after Bob has obtained a word u this way, he sets $w = [x'_i, u]$ and applies steps (2)–(4) to w approximately $\frac{|w|}{2}$ times, where $|w|$ is the length of w . This final step is needed to make this w (which is equal to 1 in $\hat{\Gamma}$, and therefore also in Γ') indistinguishable from $w \neq 1$, which is constructed in the same form $w = [x'_i, u]$, see below. Here x'_i is the same as in the relator $x'_i = \prod_{j=1}^M [x'_i, w_j]$ published by Alice; see Section 7.1.2. Having $w \neq 1$ in this form is needed, in turn, to foil “quotient attacks”.

When Bob wants to transmit an element not equal to 1 in Γ' , he should first choose a random word u from the commutator subgroup of the free group generated by x'_1, x'_2, \dots . To select a random word from the commutator subgroup is easy; Bob can select an arbitrary random word v first, and then adjust the exponents on the generators in v so that the exponent sum on every generator in v is 0. The length of u should be in the same range as the lengths of the words u equal to 1 in $\hat{\Gamma}$ constructed by Bob before. Then Bob lets $w = [x'_i, u]$, where x'_i is the same as in the relator $x'_i = \prod_{j=1}^M [x'_i, w_j]$ published by Alice; see Section 7.1.2. Finally, to hide u , he applies “shuffling” to w (steps (2)–(4) above) approximately $\frac{|w|}{2}$ times, where $|w|$ is the length of w .

Now we explain why a random word of sufficiently long length is not equal to 1 in Γ with overwhelming probability, provided Γ is a presentation described in the end of Section 7.1.2.

Like any other group, the group G given by the presentation Γ is a factor group $G = F/R$ of the ambient free group F generated by x_1, x_2, \dots . Therefore, to estimate the probability that a random word in x_1, x_2, \dots would not belong to R (and therefore, would not be equal to 1 in G), one should estimate the *asymptotic density* (see e.g. [146]) of the complement to R in the free group F . It makes notation simpler if one deals instead with the asymptotic density of R itself, which is

$$\rho_F(R) = \limsup_{n \rightarrow \infty} \frac{\#\{u \in R : |u| \leq n\}}{\#\{u \in F : |u| \leq n\}}.$$

Here $|u|$ denotes the usual lexicographic length of u as a word in x_1, x_2, \dots . Thus, the asymptotic density depends, in general, on a free generating set of F , but we will not go into these details here because all facts that we are going to need are independent of the choice of basis. One principal fact that we can use here is due to Woess [284]: if the group $G = F/R$ is infinite, then $\rho_F(R) = 0$. Since the

group G given by the presentation Γ is infinite (see our Section 7.1.2), this already tells us that the probability for a random word of length n in x_1, x_2, \dots not to be equal to 1 in G is approaching 1 when $n \rightarrow \infty$. However, if we want words transmitted by Bob to be of reasonable length (on the order of 100–200, say), we have to address the question of *how fast* the ratio in the definition of the asymptotic density converges to 0 if R is the normal closure of the relators described in the end of Section 7.1.2. It turns out that for *non-amenable* groups the convergence is exponentially fast; this is also due to Woess [284]. We are not going to explain here what amenable groups are; it is sufficient for us to know that small cancellation groups are not amenable (because they have free subgroups, see e.g. [177]). Thus, small cancellation groups are just fine for our purposes here: the probability for a random word of length n in x_1, x_2, \dots not to be equal to 1 in G is approaching 1 exponentially fast when $n \rightarrow \infty$.

Finally, we touch upon semantic security (see [102]) of the words transmitted by Bob. We do not give any rigorous probabilistic estimates since this would require at least defining a probability measure on an infinite group, which is a very nontrivial problem by itself (cf. [28]). Instead, we offer here an informal argument which we hope to be convincing. A nice thing about Bob’s encryption procedure is that when he selects a word $u \neq 1$, he simply selects a random word. Thus, $u \neq 1$ is indistinguishable from a random word just because it is random! Then, the element $w = [x'_i, u]$, which is transmitted by Bob, looks like it is no longer random because it is of a special form. However:

- (1) What is actually transmitted by Bob is a *word in the alphabet* x'_1, x'_2, \dots representing the element $w = [x'_i, u]$ of the group defined by $\hat{\Gamma}$. This word is *not* of the form $[x'_i, u]$ because Bob has applied a “shuffling” to w .
- (2) Given the specifics of our protocol, what really matters is that transmitted words equal to 1 in $\hat{\Gamma}$ are indistinguishable from transmitted words not equal to 1. This is why we require Bob’s elements representing 1 in $\hat{\Gamma}$ to be of the form $[x'_i, u]$ as well.

Thus, the question about semantic security of Bob’s transmissions boils down to the following question of independent interest: is a word u representing 1 in $\hat{\Gamma}$ indistinguishable from a random word (of the same length)? As we have admitted above, we do not have a rigorous proof that it is, but computer experiments show that when most of the relators in $\hat{\Gamma}$ have length at most 4, then the words u representing 1 in $\hat{\Gamma}$, obtained as described earlier in this section, pass at least the equal frequency test for 1-, 2-, and 3-letter subwords, thus making it appear likely that the answer to the question above is affirmative for such $\hat{\Gamma}$.

7.2. Public-key encryption and encryption emulation attacks

In this section we continue to explore how non-recursiveness of a decision problem (as opposed to computational hardness of a search problem) can be used in public-key cryptography. We follow the exposition in [216].

Here our focus is on the “encryption emulation” attack on the sender’s (Bob’s) transmissions. We show that Bob’s encryption can be made reasonably secure against the encryption emulation attack (see our Section 7.1.1) by computationally unbounded adversary, with one reservation: a legitimate receiver decrypts correctly with probability that can be made arbitrarily close to 1, but not equal to 1.

First we recall (from Section 7.1.1) that the encryption emulation attack would indeed work fine (at least, for computationally unbounded adversary) if the correctness of the scheme was perfect. However, if there is a gap, no matter how small (it can be easily made on the order of 10^{-200}), between 1 and the probability of correct decryption by a legitimate receiver, then this gap can be very substantially “amplified” for the adversary, thus making the probability of correct illegitimate decryption anything but overwhelming.

We emphasize at this point that in the protocol described in this section, we do not claim security against the “encryption emulation” (or any other) attack by computationally unbounded adversary on the receiver’s public key, but we only claim security against the “encryption emulation” attack on the sender’s transmission. It seems that the problem of security of the sender’s encryption algorithm is of independent interest. Of course, in applications to, say, Internet shopping or banking, both the sender’s and the receiver’s algorithms are assumed to be known to the adversary (“Kerckhoff’s assumptions”, see e.g. [190]), and the receiver’s decryption algorithms (or algorithms for obtaining public keys) are usually more vulnerable to attacks. However, in some other applications, say, to electronic signatures (not to mention non-commercial, e.g., military applications), decryption algorithms or algorithms for generating public keys (by the receiver) need not be public, whereas encryption algorithms (of the sender) always are.

Thus, we present here a protocol which is secure against the “encryption emulation” attack on the sender’s transmission by a computationally unbounded adversary who has complete information on the algorithm(s) and hardware that the sender uses for encryption. More precisely, in our protocol the sender transmits his private bit sequence by encrypting one bit at a time, and the receiver decrypts each bit correctly with probability that can be made arbitrarily close to 1, but not equal to 1. At the same time, the (computationally unbounded) adversary decrypts each bit (by emulating the sender’s encryption algorithm) correctly with probability at most $\frac{3}{4}$.

There are essentially no requirements on the sender’s computational abilities; in fact, encryption can be done by hand, which can be a big advantage in some situations; for example, a field operative can receive a public key from a command center and transmit encrypted information over the phone, without even using a computer.

Encryption. Now we describe an encryption protocol with the following features:

- (F1) Bob encrypts his secret bit by a word in a public alphabet X .
- (F2) Alice (the receiver) decrypts Bob’s transmission correctly with probability that can be made arbitrarily close to 1, but not equal to 1.
- (F3) The adversary, Eve, is assumed to have no bound on the speed of computation or on the storage space.
- (F4) Eve is assumed to have complete information on the algorithm(s) and hardware that Bob uses for encryption. However, Eve cannot predict outputs of Bob’s random numbers generator (the latter could be just coin tossing, say).
- (F5) Eve does not have information on Alice’s algorithm for obtaining public keys.

- (F6) Eve cannot decrypt Bob's secret bit correctly with probability $> \frac{3}{4}$ by emulating Bob's encryption algorithm.

Once again: here we only claim security against the “encryption emulation” attack (by a computationally unbounded adversary) on the sender's transmissions. This does not mean that the receiver's private keys in our protocol are insecure against real-life (i.e., computationally bounded) adversaries, but we just prefer to focus here on what is secure against a computationally unbounded adversary since this paradigm shift looks important to us (at least, from a theoretical point of view).

We also have to say up front that the encryption protocol that is presented in this section is probably not very suitable for commercial applications (such as Internet shopping or banking) for yet another reason: because of a large amount of work required from Alice to receive just one bit from Bob. Bob, on the other hand, may not even need a computer for encryption.

Now we are getting to the protocol description. In one round of this protocol, Bob transmits a single bit, i.e., Alice generates a new public key for each bit transmission.

- (P0) Alice publishes two group presentations by generators and defining relations:

$$\begin{aligned}\Gamma_1 &= \langle x_1, x_2, \dots, x_n \mid r_1, r_2, \dots, r_k \rangle, \\ \Gamma_2 &= \langle x_1, x_2, \dots, x_n \mid s_1, s_2, \dots, s_m \rangle.\end{aligned}$$

One of them defines the trivial group, whereas the other one defines an infinite group, but only Alice knows which one is which. In the group that is infinite, Alice should be able to efficiently solve the word problem, i.e., given a word $w = w(x_1, x_2, \dots, x_n)$, she should be able to determine whether or not $w = 1$ in that group. (There is a large and easily accessible pool of such groups, see our Section 7.1.2 for discussion.)

Bob is instructed to transmit his private bit to Alice as follows:

- (P1) In place of “1”, Bob transmits a pair of words (w_1, w_2) in the alphabet $X = \{x_1, x_2, \dots, x_n, x_1^{-1}, \dots, x_n^{-1}\}$, where w_1 is selected randomly, while w_2 is selected to be equal to 1 in the group G_2 defined by Γ_2 .
- (P2) In place of “0”, Bob transmits a pair of words (w_1, w_2) , where w_2 is selected randomly, while w_1 is selected to be equal to 1 in the group G_1 defined by Γ_1 .

Now we have to specify the algorithms that Bob should use to select his words.

Algorithm “0” (for selecting a word $v = v(x_1, \dots, x_n)$ not equal to 1 in a Γ_i) is quite simple: Bob just selects a random word by building it letter-by-letter, selecting each letter uniformly from the set $X = \{x_1, \dots, x_n, x_1^{-1}, \dots, x_n^{-1}\}$. The length of such a word should be a random integer from an interval that Bob selects up front, based on his computational abilities. In the end, Bob should cancel out all subwords of the form $x_i x_i^{-1}$ or $x_i^{-1} x_i$.

Algorithm “1” (for selecting a word $u = u(x_1, \dots, x_n)$ equal to 1 in a Γ_i) is slightly more complex. It amounts to applying a random sequence of operations of the following two kinds, starting with the empty word:

- (1) Inserting into a random place in the current word a pair hh^{-1} for a random word h .
- (2) Inserting into a random place in the current word a random conjugate $g^{-1}r_i g$ of a random defining relator r_i .

In the end, Bob should cancel out all subwords of the form $x_i x_i^{-1}$ or $x_i^{-1} x_i$. The length of the resulting word should be in the same range as the length of the output of Algorithm “0”. We do not go into more details here because all claims in this section remain valid no matter what algorithm for producing words equal to 1 is chosen, as long as it returns a word whose length is in the same range as that of the output of Algorithm “0”.

Now let us explain why the legitimate receiver (Alice) decrypts correctly with overwhelming probability. Suppose, without loss of generality, that the group G_1 is trivial, and G_2 is infinite. Then, if Alice receives a pair of words (w_1, w_2) such that $w_1 = 1$ in G_1 and $w_2 \neq 1$ in G_2 , she concludes that Bob intended to transmit a “0”. This conclusion is correct with probability 1. If Alice receives (w_1, w_2) such that $w_1 = 1$ in G_1 and $w_2 = 1$ in G_2 , she concludes that Bob intended to transmit a “1”. This conclusion is correct with probability which is close to 1, but not equal to 1 because it may happen, with probability $\varepsilon > 0$, that the random word w_2 selected by Bob is equal to 1 in G_2 . The point here is that, if G_2 is infinite, this ε is negligible and, moreover, for “most” groups G_2 this ε tends to 0 exponentially fast as the length of w_2 increases. For more precise statements, see our Section 7.1.3; here we just say that it is easy for Alice to make sure that G_2 is one of those groups.

Now we are going to discuss Eve’s attack on Bob’s transmission. Under our assumptions (F3), (F4) Eve can identify the word(s) in the transmitted pair which is/are equal to 1 in the corresponding group(s), as well as the word, if any, which is not equal to 1. Indeed, for any particular transmitted word w she can use the “encryption emulation” attack, as described in our Introduction: she emulates algorithms ‘0’ and ‘1’ over and over again, each time with fresh randomness, until the word w is obtained. Thus, Eve is building up two lists, corresponding to two algorithms above. As we have already pointed out in a similar situation in Section 7.1.1, the list that corresponds to the Algorithm “0” is useless to Eve because it is eventually going to contain *all* words in the alphabet $X = \{x_1, \dots, x_n, x_1^{-1}, \dots, x_n^{-1}\}$, with overwhelming probability. Therefore, Eve may just as well forget about this list and concentrate on the other one, that corresponds to the Algorithm “1”. Now the situation boils down to the following: if the word w appears on the list, then it is equal to 1 in the corresponding group G_i . If not, then not.

It may seem that Eve should encounter a problem detecting $w \neq 1$: how can she conclude that w does *not* appear on the list if the list is infinite (more precisely, of a priori unbounded length)? This is where our condition (F4) plays a role: if Eve has complete information on the algorithm(s) and hardware that Bob uses for encryption, then she does know a real-life bound on the size of the list.

Thus, Eve can identify the word(s) in the transmitted pair which is/are equal to 1 in the corresponding group(s), as well as the word, if any, which is not equal to 1. There are now the following possibilities:

- (1) $w_1 = 1$ in G_1 , $w_2 = 1$ in G_2 ;
- (2) $w_1 = 1$ in G_1 , $w_2 \neq 1$ in G_2 ;
- (3) $w_1 \neq 1$ in G_1 , $w_2 = 1$ in G_2 .

It is easy to see that one of the possibilities (2) or (3) cannot actually occur, depending on which group G_i is trivial. Then, the possibility (1) occurs with probability $\frac{1}{2}$ (either when Bob wants to transmit “1” and G_1 is trivial, or when Bob wants to transmit “0” and G_2 is trivial). If this possibility occurs, Eve cannot decrypt Bob’s bit correctly with probability $> \frac{1}{2}$ because she does not know which group G_i is trivial. As we have pointed out earlier in this section, a computationally unbounded Eve could find out which G_i is trivial, but we specifically consider attacks on the sender’s encryption here (cf. our condition (F5) above). We just note, in passing, that for a real-life (i.e., computationally bounded) adversary to find out which presentation Γ_i defines the trivial group is by no means easy and deserves to be a subject of separate investigation. There are many different ways to efficiently construct very complex presentations of the trivial group, some of them involving a lot of random choices. See e.g. [210] for a survey on the subject.

In any case, our claim (F6) was that Eve cannot decrypt Bob’s bit correctly with probability $> \frac{3}{4}$ by emulating Bob’s encryption algorithm, which is obviously true in this scheme since the probability for Eve to decrypt correctly is, in fact, precisely $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot 1 = \frac{3}{4}$. (Note that Eve decrypts correctly with probability 1 if either of the possibilities (2) or (3) above occurs.)

Someone may say that $\frac{3}{4}$ is a rather high probability of illegitimate decryption, even though this is just for one bit. Recall however that we are dealing with a computationally unbounded adversary, while Bob can essentially do his encryption by hand! All he needs is a generator of uniformly distributed random integers in the interval between 1 and $2n$ (the latter is the cardinality of the alphabet X). Besides, note that with the probability of correctly decrypting one bit equal to $\frac{3}{4}$, the probability to correctly decrypt, say, a credit card number of 16 decimal digits would be on the order of 10^{-7} , which is comparable to the chance of winning the jackpot in a lottery. Of course, there are many tricks that can make this probability much smaller, but we think we should stop here because, as we have pointed out before, our focus here is on the new paradigm itself.

CHAPTER 8

Authentication

The main goal of an authentication protocol is to allow a legitimate user (Alice) to prove her identity over an insecure channel to a server (Bob) using her (long-term) private key. Of particular interest are *zero-knowledge* proofs of identity, which means that if the identity is true, no malicious verifier learns anything other than this fact.

We only consider public-key authentication protocols here, i.e., we assume that there is no pre-existing arrangement. Alice is usually referred to as the *prover* and Bob is referred to as the *verifier*. For a general theory of public-key authentication (a.k.a. identification) as well as early examples of authentication protocols, the reader is referred to [190].

The problem of authentication is, arguably, the most intriguing one in public-key cryptography since it is, in a way, “between” the two principal problems: (1) the existence of one-way functions; (2) the existence of a secure public-key cryptosystem (or, equivalently, the existence of a secure public key establishment protocol). Namely, the existence of a secure public-key cryptosystem implies the existence of a secure public-key authentication scheme, and the latter implies the existence of one-way functions. However, it is unknown whether or not the reverse implications hold. We describe in this chapter a number of authentication schemes based on different ideas, some of them coming from group theory, some from other areas of mathematics.

There are several general proposals available that use non-abelian groups; some of them are direct adaptations of key exchange schemes such as, e.g., the Ko-Lee key exchange protocol; in particular, they are based on the conjugacy search problem. These are discussed in Sections 8.1, 8.2, 8.3, 8.4. Then, in sections 8.5 and 8.6, we describe more recent proposals following joint work of Grigoriev and Shpilrain.

8.1. A Diffie-Hellman-like scheme

Let G be a group and $A, B < G$ two commuting subgroups of G , i.e., $ab = ba$ for any $a \in A$ and $b \in B$.

Alice’s private key is an element $s \in A$. Alice’s public key is a pair (w, t) , where w is an arbitrary element of G and $t = s^{-1}ws$. The authentication protocol is the following sequence of transactions:

- (1) Bob chooses $r \in B$ and sends a challenge $w' = r^{-1}wr$ to Alice.
- (2) Alice sends the response $w'' = s^{-1}w's$ to Bob.
- (3) Bob checks if $w'' = r^{-1}tr$.

A correct answer of the prover at the second step leads to acceptance by the verifier because by design the elements r and s commute, and hence the equality

$$w'' = s^{-1}w's = s^{-1}r^{-1}wrs = r^{-1}s^{-1}wsr = r^{-1}tr$$

is satisfied.

The intruder (Eve) who wants to authenticate as Alice to Bob can do either of the following:

- **Compute Alice's private key s** by solving the conjugacy search problem relative to the subgroup A or by solving the decomposition problem for a pair (w, t) relative to a subgroup A , i.e., computing elements $s_1, s_2 \in A$ such that $t = s_1ws_2$. It is trivial to check that for such a pair (s_1, s_2) the equality $s_1w's_2 = w''$ is always satisfied and hence the pair (s_1, s_2) works as Alice's private key s .
- **Compute Bob's temporary key r** by solving the conjugacy search problem relative to the subgroup B or by solving the decomposition problem for a pair (w, w') relative to the subgroup B and use it as described above for attack on Alice's private key.

The computational hardness of these problems depends on particular subgroups A and B , so it does not have to be the same for A and B .

REMARK 8.1.1. This scheme can be easily modified to be based on the decomposition problem. Indeed, suppose that A_1, A_2, B_1, B_2 are subgroups of G such that $[A_1, B_1] = 1$ and $[A_2, B_2] = 1$. Alice's private key is a pair $(a_1, a_2) \in A_1 \times A_2$, and her public key is a pair $(w, t) \in G \times G$, where w is an arbitrary element of G and $t = a_1wa_2$. Now the transactions are modified as follows:

- Bob chooses elements $b_1 \in B_1, b_2 \in B_2$ and sends a challenge $w' = b_1wb_2$ to Alice.
- Alice sends the response $w'' = a_1w'a_2$ to Bob.
- Bob checks if $w'' = b_1tb_2$.

This scheme is due to Lal and Chaturwedi [166]. It seems that the conjugacy search problem cannot be used to attack this modification of the protocol.

8.2. A Feige-Fiat-Shamir-like scheme

Another proposal of a group based authentication scheme is due to Sibert et al. [256]. This scheme is reminiscent of the Feige-Fiat-Shamir scheme [74] that involves repeating several times a three-pass challenge-response step. One of the important differences with the Diffie-Hellman-like scheme described above is that there is no need to choose commuting subgroups A and B .

As above, Alice's private key is an element $s \in G$ and her public key is a pair (w, t) , where w is an arbitrary element of G and $t = s^{-1}ws$. The authentication protocol goes as follows:

- (1) Alice chooses a random $r \in G$ and sends the element $x = r^{-1}tr$, called the *commitment*, to Bob.
- (2) Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends $y = r$ to Bob and Bob checks if the equality $x = y^{-1}ty$ is satisfied.
 - If $c = 1$, then Alice sends $y = sr$ to Bob and Bob checks if the equality $x = y^{-1}wy$ is satisfied.

It is trivial to check that a correct answer of the prover at the second step leads to acceptance by the verifier. It is somewhat less obvious why such an arrangement (with a random bit) is needed; it may seem that Alice could just reveal $y = sr$: this does not reveal the secret s , and yet allows Bob to verify the equality $x = y^{-1}wy$. The point is that in this case, the adversary, Eve, who wants to impersonate Alice can just take an arbitrary element u and send $x = u^{-1}wu$ to Bob as a commitment. Then this u would play the same role as $y = sr$ for verification purposes. Similarly, if Eve knew for sure that Alice would send $y = r$ for verification purposes, she would use an arbitrary element u in place of r at the commitment step. These considerations also show that the above authentication protocol should be run several times for better reliability because with a single run, Eve can successfully impersonate Alice with probability $\frac{1}{2}$. After k runs, this probability goes down to $\frac{1}{2^k}$.

Similarly to the Diffie-Hellman-like scheme, the security of the Fiat-Shamir-like scheme depends on the computational hardness of the conjugacy search problem in the group G . It is interesting to note that the decomposition search problem cannot be used to attack this particular protocol because Bob accepts exactly one element, either r or sr , depending on the value of c , at the last step of the protocol.

8.3. Authentication based on the twisted conjugacy problem

The authentication scheme described in the previous section can be modified to be based on the decomposition problem in a similar way as that was done in Section 8.1. This scheme also allows a more interesting modification. Let φ be an arbitrary endomorphism (i.e., homomorphism into itself) of the platform group G . We assume that φ is publicly known, e.g., it can be part of Alice's public key. Alice's private key is an element $s \in G$, and her public key is a pair (w, t) , where w is an arbitrary element of G and $t = s^{-1}w\varphi(s)$. The authentication protocol goes as follows:

- Alice selects an element $r \in G$ and sends the element $x = r^{-1}t\varphi(r)$, called the *commitment*, to Bob.
- Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends $y = r$ to Bob and Bob checks if the equality $x = y^{-1}t\varphi(y)$ is satisfied.
 - If $c = 1$, then Alice sends $y = sr$ to Bob and Bob checks if the equality $x = y^{-1}w\varphi(y)$ is satisfied.

Again, a correct answer of the prover at the second step leads to acceptance by the verifier.

To break the protocol it is sufficient to find any element $s' \in G$ such that $t = (s')^{-1}w\varphi(s')$ which is an instance of the problem known as the *twisted conjugacy (search) problem*:

Let G be a group. For any $\varphi \in Aut(G)$ and a pair of elements $w, t \in G$ find a *twisted conjugator* for w and t , i.e., an element $s \in G$ such that $t = s^{-1}w\varphi(s)$ provided at least one such s exists.

The decision version of this problem is a relatively new algorithmic problem in group theory; it is very non-trivial even for free groups; see [26]. Another class of groups where this problem was considered is the class of polycyclic-by-finite groups [76].

8.4. Authentication from matrix conjugation

In this and the following two sections, we describe recent proposals of authentication schemes due to Grigoriev and Shpilrain. In this section, our exposition follows [115]. Here again, the conjugacy search problem plays a role, but there is a remarkable novelty in the verification procedure; in particular, it is randomized, which creates additional difficulties for the adversary. We think it makes sense to spell out what makes the proposal in this section different from other proposals based on the conjugacy search problem:

- (1) Forgery (a.k.a. impersonation) seems infeasible without finding the prover’s private key. In other proposals, there is usually a “shortcut”, i.e., a way for the adversary to pass the final test by the verifier without obtaining the prover’s private key. In particular, in the proposal modeled on the Diffie-Hellman authentication scheme (see our Section 8.1), there is an alternative (formally weaker) problem that is sufficient for the adversary to solve in order to impersonate the prover. Namely, it is sufficient for the adversary to obtain $Y^{-1}X^{-1}AXY$ from $X^{-1}AX$, $Y^{-1}AY$, and A .
- (2) Our platform semigroup might be the first serious candidate for having a generically hard conjugacy search problem. It can therefore be used with some other previously suggested cryptographic protocols based on the conjugacy search problem.
- (3) One of the most important new features is that the verifier *selects his final test randomly from a large series of tests*. This is what makes it difficult for the adversary to impersonate the prover without obtaining her private key: if the adversary just “studies for the test”, as weak students do, he/she at least should know what the test is.
- (4) Unlike the proposals in [116, 256], our authentication scheme does not use the Feige-Fiat-Shamir idea [74] involving repeating, several times, a three-pass challenge-response step (to avoid predicting, by the adversary, the challenge with non-negligible probability). In our scheme, we have just one challenge and one response.
- (5) To prevent attacks by a malicious verifier, there is an intermediate “commitment to challenge” step for the verifier because otherwise, a malicious verifier might present the prover with a carefully selected challenge that may result in leaking information about the prover’s private key at the response step. This is similar to the “chosen-plaintext attack” on an encryption protocol.

8.4.1. The protocol, beta version. We start by giving a preliminary description of our authentication protocol. Here, as usual, Alice is the prover and Bob the verifier. We call this a “beta version” of the protocol because what we describe here represents a single session; repeating this particular protocol several times can compromise the long-term private key of the prover. This is why extra care has to be taken to protect the long-term private key; this is done in the complete protocol described in the following section, while here, in an attempt to be helpful to the reader, we describe the “skeleton” of our scheme where all principal (i.e., non-technical) ideas are introduced.

The platform ring G that we suggest is the ring of $n \times n$ matrices over N -truncated k -variable polynomials over a ring R . The reader is referred to [115]

for the definition of N -truncated polynomials as well as for suggested values of parameters n , N , k , and the ring R .

Protocol, beta version

- (i) Alice's public key is a pair of matrices $(A, X^{-1}AX)$, where the matrix $X \in G$ is Alice's long-term private key. The matrix $A \in G$ does not have to be invertible.
- (ii) At the challenge step, Bob chooses a random matrix B from the ring G and sends it to Alice. (See the full version of the protocol in the next section for how to prevent Bob from choosing B maliciously.)
- (iii) Alice responds with the matrix $X^{-1}BX$.
- (iv) Bob selects a random word $w(x, y)$ (without negative exponents on x or y), evaluates the matrices $M_1 = w(A, B)$ and $M_2 = w(X^{-1}AX, X^{-1}BX)$, then computes their traces. If $\text{tr}(M_1) = \text{tr}(M_2)$, he accepts authentication. If not, then rejects.

The point of the final test is that $M_2 = w(X^{-1}AX, X^{-1}BX)$ should be equal to $X^{-1}M_1X = X^{-1}w(A, B)X$. Therefore, since the matrices M_1 and M_2 are conjugate, they should, in particular, have the same trace. Note that the trace in this context works much better (from the security point of view) than, say, the determinant, because the determinant is a multiplicative function, so the adversary could use any matrix with the same determinant as B in place of $X^{-1}BX$, and still pass the determinant test. With the trace, the situation is quite different, and there is no visible way for the adversary to pass the trace test for a random word $w(x, y)$ unless he/she actually uses the matrix $X^{-1}BX$.

8.4.2. The protocol, full version. Compared to the beta version described in the previous section, the full protocol given in this section has an extra feature of protecting the long-term private key X from overexposure. This is needed because upon accumulating sufficiently many matrices of the form $X^{-1}B_iX$ with different B_i but the same X , the adversary may recover X more easily. To avoid this, we make Alice (the prover) apply a non-invertible endomorphism (i.e., a homomorphism into itself) of the ambient ring G to all participating matrices. This endomorphism is selected by Bob in the beginning of each new session. We also note yet another extra feature of the protocol below, namely, a (mild) "commitment to challenge" by the verifier (step 2i) preceding the actual challenge. Recall that this is done to prevent a malicious verifier from presenting the prover with a carefully selected challenge that may result in leaking information about the prover's private key at the response step.

Protocol, full version

- (1) Alice's public key is a pair of matrices $(A, X^{-1}AX)$, where the matrix $X \in G$ is Alice's long-term private key. The matrix $A \in G$ does not have to be invertible.
- (2) At the "commitment to challenge" step, Bob chooses: (i) a random matrix B from the ring G ; (ii) a random non-invertible endomorphism φ of the ring G . Bob then sends B and φ to Alice.
- (3) In order to prevent a malicious Bob from presenting her with a carefully selected challenge, Alice publishes random positive integers p and q and

- asks Bob to send her random non-zero constants $c_i, i = 1, 2, 3$, and create his challenge in the form $B' = c_1 A + c_2 B + c_3 A^p B^q$.
- (4) Upon receiving B' , Alice responds with the matrix $\varphi(X^{-1}B'X)$.
 - (5) Bob selects a random word $w(x, y)$ (without negative exponents on x or y), evaluates the matrices $M_1 = w(\varphi(A), \varphi(B'))$ and $M_2 = w(\varphi(X^{-1}AX), \varphi(X^{-1}B'X))$, then computes their traces. If $\text{tr}(M_1) = \text{tr}(M_2)$, he accepts authentication. If not, then rejects.

8.4.3. Cryptanalysis. We start by discussing how the adversary, Eve, can attack Alice’s long-term private key (the matrix X) directly, based just on the public key $P = X^{-1}AX$. The relevant problem is known as the *conjugacy search problem*. Note that the equation $P = X^{-1}AX$ implies $XP = AX$, which translates into a system of n^2 linear equations for the entries of X , where n is the size of participating matrices. Thus, a natural way for Eve to attempt to find X would be to solve this system. However, there are some major obstacles along this way:

- (a) The matrix equation $XP = AX$ is *not equivalent* to $P = X^{-1}AX$. The former equation has many solutions; for example, if X is a solution, then any matrix of the form $X' = f(A) \cdot X \cdot g(P)$ is a solution, too, where $f(A)$ and $g(P)$ are arbitrary polynomials in the matrices A and P , respectively. However, only *invertible* matrices X' will be solutions of the equation $P = X^{-1}AX$. If participating matrices come from a ring where “generic” matrices are non-invertible (which is the case for our suggested platform ring), then Eve would have to add to the matrix equation $XP = AX$ another equation $XY = I$, where X, Y are unknown matrices, and I is the identity matrix. This translates into a system of n^2 *quadratic* equations, not linear ones.
- (b) As explained in the previous paragraph, Eve is facing a system of n^2 linear equations and n^2 quadratic equations, with $2n^2$ unknowns, over a ring R , which in our scheme is the ring of N -truncated k -variable polynomials over \mathbf{Z}_{11} . She can further translate this into a system of linear equations over \mathbf{Z}_{11} if she collects coefficients at similar monomials, but this system is going to be huge: as explained in [115], it is going to have more than 10^{20} equations (by the number of monomials). Note that, although entries of all participating matrices are sparse polynomials, Eve does not know which monomials in the private matrix X occur with non-zero coefficients, which means she has to either engage *all* monomials in her equations or try all possible supports (i.e., collections of monomials with non-zero coefficients) of the entries of elementary matrices in a decomposition of X .
- (c) Eve may hope to get more information about the matrix X if she eavesdrops on several authentication sessions between legitimate parties. More specifically, she can accumulate several pairs of matrices of the form $(\varphi_i(B_i), \varphi_i(X^{-1}B_iX))$. Note however that even if a pair like that yields some information, this is going to be information about the matrix $\varphi_i(X)$ rather than about X itself. To recover X from $\varphi_i(X)$ is impossible because φ_i has a large kernel by design.

8.5. Authentication from actions on graphs, groups, or rings

In this section, our exposition follows [116]. Here we propose several general Feige-Fiat-Shamir-like [74] constructions of authentication schemes (with long-term private keys) from arbitrary actions.

Suppose a (partial) semigroup S acts on a set X , i.e., for $s, t \in S$ and $x \in X$, one has $(st)(x) = s(t(x))$ whenever both sides are defined. For cryptographic purposes, it is good to have an action which is “hard-to-invert”. We deliberately avoid using the “one-way function” terminology here because we do not want to be distracted by formal definitions that are outside of the main focus of this book. For a rigorous definition of a one-way function, we just refer to one of the well-established sources, such as [99]. It is sufficient for our purposes to use an intuitive idea of a hard-to-invert action which is as follows. Let X and Y be two sets such that complexity $|u|$ is defined for all elements u of either set. A function $f : X \rightarrow Y$ is hard-to-invert if computing $f(x)$ takes time polynomial in $|x|$ for any $x \in X$ (which implies, in particular, that complexity of $f(x)$ is bounded by a polynomial function of $|x|$), but there is no known algorithm that would compute some $f^{-1}(y)$ in polynomial time in $|y|$ for every $y \in f(X)$.

In our context of actions, we typically consider hard-to-invert functions of the type $f_x : s \rightarrow s(x)$; in particular, a secret is usually a *mapping* s , which makes our approach different from what was considered before. This idea allows us to construct several general Feige-Fiat-Shamir-like authentication schemes (with long-term private keys) from arbitrary actions; see Section 8.5.2. Then, in the subsequent sections, we give several concrete realizations of this general idea, and in particular, we describe several authentication schemes where recovering the prover’s long-term private key from her public key is an NP-hard problem. We note however that what really matters for cryptographic security is computational intractability of a problem on a *generic* set of inputs, i.e., the problem should be hard on “most” randomly selected inputs. Some of the problems that we employ in this section, e.g., Graph Colorability, are *likely* to be generically NP-hard, which makes them quite attractive for cryptographic applications.

We also address an apparently easier task of *forgery* (a.k.a. *misrepresentation*, a.k.a. *impersonation*), and show that in most of our schemes this, too, is equivalent for the adversary to solving an NP-hard problem. To be more specific, by *forgery* we mean the scenario where the adversary enters the authentication process at the *commitment* step, and then has to respond to the *challenge* properly.

8.5.1. When a composition of functions is hard-to-invert. Here we prove a simple but useful proposition about composing a hard-to-invert function with another function, which is not necessarily hard-to-invert.

PROPOSITION 8.5.1. *Let A, B and C be sets, and let $\varphi : A \rightarrow B$ and $\psi : B \rightarrow C$ be two functions such that computing $\varphi(x)$ as well as $\psi(x)$ takes time polynomial in $|x|$ for any x for which the corresponding function is defined.*

- (a) *If ψ is hard-to-invert and the domain of ψ is contained in the range of φ , then the composition $\varphi\psi = \psi(\varphi)$ is hard-to-invert.*
- (b) *If φ is hard-to-invert, ψ is injective (i.e., one-to-one), and the domain of ψ contains the range of φ , then the composition $\varphi\psi = \psi(\varphi)$ is hard-to-invert.*

Proof. (a) Let $f = \psi(\varphi) : A \rightarrow C$. By way of contradiction, suppose there is an algorithm \mathcal{A} that computes some $f^{-1}(c)$ in polynomial time in $|c|$ for every $c \in f(A)$.

Now let $y \in \psi(B)$. Since the domain of ψ is contained in the range of φ , this implies $y \in f(A)$. Then we apply the algorithm \mathcal{A} to y to get some $a \in A$. This takes polynomial time in $|y|$, and, in particular, the size of a is polynomial in $|y|$. Then we apply φ to a to get an element $b \in B$. This takes polynomial time in $|a|$, and therefore also in $|y|$. Since now $\psi(b) = y$, we have found a preimage of y under ψ in polynomial time in $|y|$, contradicting the assumption on ψ to be hard-to-invert.

(b) Again, let $f = \psi(\varphi) : A \rightarrow C$ and suppose, by way of contradiction, that there is an algorithm \mathcal{A}_1 that computes some $f^{-1}(c)$ in polynomial time in $|c|$ for every $c \in f(A)$.

Now let $y \in \varphi(A)$. Since the domain of ψ contains the range of φ , we can apply ψ to y to get $\psi(y) = c \in C$. This takes polynomial time in $|y|$. Then we apply the algorithm \mathcal{A}_1 to c to obtain some $a = f^{-1}(c)$. This takes polynomial time in $|c|$, and therefore also in $|y|$. Now we claim that $\varphi(a) = y$, for if it was not the case, we would have $y_1 \neq y$ such that $\psi(y) = \psi(y_1) = c$ (since $f(a)$ should be equal to c), contradicting the assumption on ψ to be injective. ■

8.5.2. Three protocols. In this section, we give a description of three generic authentication protocols (or, rather, “meta-protocols”, or “primitives”, since we do not give any implementation details in this section). As usual, Alice is the prover and Bob the verifier.

Protocol I. Suppose a set S acts on a set X , i.e., for any $s \in S$ and $x \in X$, the element $s(x) \in X$ is well-defined.

- (1) Alice’s public key consists of a set X , a (partial) semigroup S , an element $x \in X$, and an element $u = s(x)$ for some randomly selected $s \in S$, which is her long-term private key.
- (2) To begin authentication, Alice selects an element $t \in S$ and sends the element $v = t(s(x)) \in X$, called the *commitment*, to Bob.
- (3) Bob chooses a random bit c , called the *challenge*, and sends it to Alice.
 - If $c = 0$, then Alice sends the element t to Bob, and Bob checks if the equality $v = t(u)$ is satisfied. If it is, then Bob accepts the authentication.
 - If $c = 1$, then Alice sends the composition ts to Bob, and Bob checks if the equality $v = ts(x)$ is satisfied. If it is, then Bob accepts the authentication.

Protocol II. Yet another protocol involving a composition of actions (or mappings) is as follows.

- (1) Alice’s public key consists of a set X , a (partial) semigroup S whose elements may act on X , an element $x \in X$, and an element $z = r(x)$ for some randomly selected $r \in S$, which is her long-term private key.
- (2) To begin authentication, Alice selects an element $y \in X$, together with two elements $s, t \in S$ such that $s(x) = y$ and $t(y) = z$. She then sends the element y (the commitment) to Bob.
- (3) Bob chooses a random bit c , the challenge, and sends it to Alice.

- If $c = 0$, then Alice sends the element s to Bob, and Bob checks if the equality $s(x) = y$ is satisfied. If it is, then Bob accepts the authentication.
- If $c = 1$, then Alice sends the element t to Bob, and Bob checks if the equality $t(y) = z$ is satisfied. If it is, then Bob accepts the authentication.

PROPOSITION 8.5.2. *Suppose that after several runs of steps (2)-(3) of the above Protocol II, both values of c are encountered. Then successful forgery in such a protocol is equivalent to compromising Alice's long-term private key, i.e., to finding $r' \in S$ such that $z = r'(x)$.*

Proof. Suppose Eve wants to impersonate Alice. To that effect, she interferes with the commitment step by sending her own commitment $y' \in X$ to Bob, such that $s'(x) = y'$ and $t'(y') = z$ for some $s', t' \in S$. Since she should be prepared to respond to both challenges $c = 0$ and $c = 1$, she should be able to produce s' as well as t' . Therefore, she is able to also produce their composition $t's'$. The result now follows from: $z = t'(y') = t'(s'(x)) = (t's')(x)$, so that $r' = t's'$. ■

Protocol III. In this protocol, the hardness of obtaining the long-term private key for the adversary can be based on "most any" search problem; we give some concrete examples in the following subsections, whereas here we give a generic protocol.

- (1) Alice's public key consists of a set S that has a property \mathcal{P} . Her long-term private key is a *proof* (or a "witness") that S does have this property. We are also assuming that the property \mathcal{P} is preserved by *isomorphisms*.
- (2) To begin authentication, Alice selects an isomorphism $\varphi : S \rightarrow S_1$ and sends the set S_1 (the commitment) to Bob.
- (3) Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends the isomorphism φ to Bob, and Bob checks:
 - (i) if $\varphi(S) = S_1$ and (ii) if φ is an isomorphism.
 - If $c = 1$, then Alice sends a proof of the fact that S_1 has the property \mathcal{P} to Bob, and Bob checks its validity.

The following proposition says that in the Protocol III, successful forgery is equivalent for the adversary to finding Alice's private key from her public key, which is equivalent, in turn, to giving a proof (or a "witness") that S does have the property \mathcal{P} . The latter problem can be selected from a large pool of NP-hard problems (see e.g. [89]).

PROPOSITION 8.5.3. *Suppose that after several runs of steps (2)-(3) of the above Protocol III, both values of c are encountered. Then successful forgery in such a protocol is equivalent to finding a proof of the fact that S has the property \mathcal{P} .*

Proof. Suppose Eve wants to impersonate Alice. To that effect, she interferes with the commitment step by sending her own commitment S'_1 to Bob. Since she should be prepared to respond to the challenge $c = 0$, she should know an isomorphism $\varphi' : S \rightarrow S'_1$. On the other hand, since she should be prepared for the challenge $c = 1$, she should know a proof of the fact that S'_1 has the property \mathcal{P} . Therefore, since φ' is invertible, this implies that she can produce a proof of the fact that S has the property \mathcal{P} . This completes the proof in one direction. The other direction is trivial. ■

REMARK 8.5.4. We note that finding a proof of the fact that a given S has a property \mathcal{P} is not a decision problem, but rather a search problem, so we cannot formally allocate it to one of the established complexity classes. However, we observe that, if there were an algorithm \mathcal{A} that would produce, for any S having a property \mathcal{P} , a proof of that fact in time bounded by a polynomial $P(|S|)$ in the “size” $|S|$ of S , then, given an arbitrary S' , we could run the algorithm \mathcal{A} on S' , and if it would not produce a proof of S' having the property \mathcal{P} after running over the time $P(|S'|)$, we could conclude that S' does not have the property \mathcal{P} , thereby solving the corresponding decision problem in polynomial time.

8.5.3. Subgraph isomorphism. There is a classical realization of the Protocol I from Section 8.5.2 (actually, it also fits in with the Protocol III), based on the Graph Isomorphism problem, see [101]. We note that this *decision* problem is in the class NP, but it is not known to be NP-hard. Moreover, generic instances of this problem are easy, because two random graphs are typically non-isomorphic for trivial reasons. However, the problem that is actually used in [101] is a *promise* problem: given two isomorphic graphs, find a particular isomorphism between them. This is not a decision problem; therefore, if we are to allocate it to one of the established complexity classes, we need some kind of “stratification” to convert it to a decision problem. This can be done as follows. Any isomorphism of a graph Γ on n vertices can be identified with a permutation of the tuple $(1, 2, \dots, n)$, i.e., with an element of the symmetric group S_n . If we choose a set of generators $\{g_i\}$ of S_n , we can ask whether or not there is an isomorphism between two given graphs Γ and Γ_1 , which can be represented as a product of at most k generators g_i . To the best of our knowledge, the question of NP-hardness of this problem has not been addressed in the literature, but it looks like a really interesting and important problem.

In this section, we describe a realization of the Protocol II from Section 8.5.2, based on the Subgraph Isomorphism problem. It is very similar to the Graph Isomorphism problem, but unlike the Graph Isomorphism problem, it is *known* to be NP-hard; see e.g. [89, Problem GT48]. We also note that this problem contains many other problems about graphs, including the Hamiltonian Circuit problem, as special cases. The Subgraph Isomorphism problem is: given two graphs Γ_1 and Γ_2 , find out whether or not Γ_1 is isomorphic to a subgraph of Γ_2 .

- (1) Alice’s public key consists of two graphs, Γ and Γ_2 . Alice’s private key is a subgraph Γ_1 of Γ_2 and an isomorphism $\varphi : \Gamma \rightarrow \Gamma_1$.
- (2) To begin authentication, Alice selects an “intermediate” graph Λ , which is a subgraph of Γ_2 , and an isomorphic embedding $\psi : \Gamma \rightarrow \Lambda$, with $\psi(\Gamma) = \Gamma_1$. Then she sends the graph Λ (the commitment) to Bob, while keeping the embeddings $\psi : \Gamma \rightarrow \Lambda$ and $\tau : \Lambda \rightarrow \Gamma_2$ to herself.
- (3) Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends the embedding ψ to Bob, and Bob checks if ψ is actually an embedding of Γ into Λ .
 - If $c = 1$, then Alice sends the embedding τ to Bob, and Bob checks if τ is actually an embedding of Λ into Γ_2 .

We point out here the following corollary to our Proposition 8.5.2:

COROLLARY 8.5.5. *Suppose that after several runs of steps (2)-(3) of the above protocol, both values of c are encountered. Then successful forgery in such a protocol is equivalent to compromising Alice’s long-term private key, i.e., to finding an embedding φ' of Γ into Γ_2 .*

We note that the problem alluded to at the end of this corollary (the Subgraph Isomorphism problem) is NP-complete; see e.g. [89, Problem GT48].

A few more comments are in order.

- As it is usual with Feige-Fiat-Shamir-like authentication protocols, steps (2)-(3) of this protocol have to be iterated several times to prevent a successful forgery with non-negligible probability.
- When we say that Alice “sends” (or “publishes”) a graph, that means that Alice sends or publishes its adjacency matrix. Thus, the size of Alice’s public key is roughly $2n^2$, where n is the number of vertices in Γ .
- When we say that Alice “sends a subgraph” of a bigger graph, that means that Alice sends the numbers $\{m_1, m_2, \dots, m_n\}$ of vertices that define this subgraph in the bigger graph. When she sends such a subgraph together with an isomorphism from another (sub)graph, she sends a map $(k_1, k_2, \dots, k_n) \rightarrow (m_1, m_2, \dots, m_n)$ between the vertices.
- Alice can construct the “intermediate” graph Λ at Step 2 of the protocol by simply discarding some randomly selected those vertices (together with incident edges) of the graph Γ_2 that do not belong to Γ_1 . Since Alice knows an embedding of Γ into Γ_2 , she will then know an embedding of Γ into Λ , too.

8.5.4. Graph homomorphism. In this section, we use the Graph Homomorphism problem that is known to be NP-complete; see [89, Problem GT52]. We have to briefly describe this problem first.

Given two graphs, Γ_1 and Γ_2 , the Graph Homomorphism problem asks whether or not there is a homomorphism $f : \Gamma_1 \rightarrow \Gamma_2$, i.e., a mapping from the vertex set of Γ_1 onto the vertex set of Γ_2 such that for any two adjacent vertices v_1, v_2 of Γ_1 , their images $f(v_1)$ and $f(v_2)$ are adjacent in Γ_2 . We note that the Graph Homomorphism problem remains NP-complete even if Γ_2 is a triangle; see [89, Problem GT52].

Now the authentication protocol is as follows.

- (1) Alice’s public key consists of two graphs, Γ_1 and Γ_2 . Alice’s long-term private key is a homomorphism $\alpha : \Gamma_1 \rightarrow \Gamma_2$.
- (2) To begin authentication, Alice selects a graph Γ together with a homomorphism $\beta : \Gamma \rightarrow \Gamma_1$ and sends the graph Γ (the commitment) to Bob, while keeping β to herself.
- (3) Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends the homomorphism β to Bob, and Bob checks whether $\beta(\Gamma) = \Gamma_1$ and whether β is a homomorphism (i.e., whether β takes adjacent vertices to adjacent ones).
 - If $c = 1$, then Alice sends the composition $\alpha\beta = \beta(\alpha)$ to Bob, and Bob checks whether $\alpha\beta(\Gamma) = \Gamma_2$ and whether $\alpha\beta$ is a homomorphism.

We now give a couple of comments on the above protocol.

- To generate her public key, Alice starts with a random graph Γ_2 and constructs Γ_1 as follows. She selects randomly a subset V' of the vertex set of Γ_2 , and for each vertex v from V' does the following. First, she

replaces v by several new vertices u_1, \dots, u_k . These vertices are going to be mapped onto the vertex v by the homomorphism that Alice is trying to construct. Thus, Alice arbitrarily connects each u_i to some other vertices adjacent to v . She repeats this procedure with each vertex from V' and obtains her private homomorphism α as a composition of intermediate homomorphisms.

The same way Alice can construct a graph Γ from Γ_1 at the commitment step.

- We note that, instead of trying to find a homomorphism between given graphs, Eve can try to find any graph Γ' that would map homomorphically onto both Γ_1 and Γ_2 , together with the corresponding homomorphisms. Then she can interfere at the commitment step and send this Γ' to Bob, which will allow her to respond to either challenge by Bob successfully. The problem of finding such a graph Γ' (a “common multiple” of two given graphs, so to speak) is of independent interest. We do not know whether it has been previously addressed in the literature.

8.5.5. Graph colorability. Graph colorability (more precisely, k -colorability) appears as problem [GT4] on the list of NP-complete problems in [89]. We include an authentication protocol based on this problem here as a special case of the Protocol III from Section 8.5.2. We note that a (rather peculiar) variant of this problem was shown to be NP-hard *on average* in [275] (the latter paper deals with edge coloring though). As we have pointed out in our Section 8.5.2, “most any” search problem can be used in Protocol III; we choose the graph colorability problem here just to illustrate this point, i.e., we do not claim that this is the best choice of underlying problem in terms of security, say.

- (1) Alice’s public key is a k -colorable graph Γ , and her private key is a k -coloring of Γ , for some (public) k .
- (2) To begin authentication, Alice selects an isomorphism $\psi : \Gamma \rightarrow \Gamma_1$, and sends the graph Γ_1 (the commitment) to Bob.
- (3) Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends the isomorphism ψ to Bob. Bob verifies that ψ is, indeed, an isomorphism from Γ onto Γ_1 .
 - If $c = 1$, then Alice sends a k -coloring of Γ_1 to Bob. Bob verifies that this is, indeed, a k -coloring of Γ_1 .

Again, a couple of comments are in order.

- It is obvious that if Γ is k -colorable and Γ_1 is isomorphic to Γ , then Γ_1 is k -colorable, too.
- When we say that Alice “sends a k -coloring”, that means that Alice sends a set of pairs (v_i, n_i) , where v_i is a vertex and n_i are integers between 1 and k such that, if v_i is adjacent to v_j , then $n_i \neq n_j$.
- Alice’s algorithm for creating her public key (i.e., a k -colorable graph Γ) is as follows. First she selects a number n of vertices; then she partitions n into a sum of k positive integers: $n = n_1 + \dots + n_k$. Now the vertex set V of the graph Γ will be the union of the sets V_i of cardinality n_i . No two vertices that belong to the same V_i will be adjacent, and any two vertices that belong to different V_i will be adjacent with probability $\frac{1}{2}$.

The k -coloring of Γ is then obvious: all vertices in the set V_i are colored in color i .

PROPOSITION 8.5.6. *Suppose that after several runs of steps (2)-(3) of the above protocol, both values of c are encountered. Then successful forgery is equivalent to finding a k -coloring of Γ .*

Proof. Suppose Eve wants to impersonate Alice. To that effect, she interferes with the commitment step by sending her own commitment Γ'_1 to Bob. Since she should be prepared to respond to the challenge $c = 0$, she should know an isomorphism ψ' between Γ and Γ'_1 . On the other hand, since she should be prepared for the challenge $c = 1$, she should be able to produce a k -coloring of Γ_1 . Since she knows ψ' and since ψ' is invertible, this implies that she can produce a k -coloring of Γ . This completes the proof in one direction. ■

The other direction is trivial. ■

8.5.6. Endomorphisms of groups or rings. In this section, we describe a realization of the Protocol I from Section 8.5.2 based on an algebraic problem known as the *endomorphism problem*, which can be formulated as follows. Given a group (or a semigroup, or a ring, or whatever) G and two elements $g, h \in G$, find out whether or not there is an endomorphism of G (i.e., a homomorphism of G into itself) that takes g to h .

For some particular groups (and rings), the endomorphism problem is known to be equivalent to the Diophantine problem (see [235, 236]), and therefore the decision problem in these groups is algorithmically unsolvable, which implies that the related search problem does not admit a solution in time bounded by any recursive function of the size of an input.

We also note at this point that there is evidence (see e.g. [238]) that finding an isomorphism (or a nontrivial homomorphism) between (finite-dimensional) *algebras* over \mathbf{Q} is hard.

Below we give a description of the authentication protocol based on the endomorphism problem, without specifying a platform group (or a ring), and then discuss possible platforms.

- (1) Alice's public key consists of a group (or a ring) G and two elements $g, h \in G$, such that $\varphi(g) = h$ for some endomorphism $\varphi \in \text{End}(G)$. This φ is Alice's private key.
- (2) To begin authentication, Alice selects an automorphism ψ of G and sends the element $v = \psi(h)$ (the commitment) to Bob.
- (3) Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends the automorphism ψ to Bob, and Bob checks whether $v = \psi(h)$ and whether ψ is an automorphism.
 - If $c = 1$, then Alice sends the composite endomorphism $\psi\varphi = \psi(\varphi)$ to Bob, and Bob checks whether $\psi\varphi(g) = v$ and whether $\psi\varphi$ is an endomorphism.

Here we point out that checking whether a given map is an endomorphism (or an automorphism) depends on how the platform group G is given. If, for example, G is given by generators and defining relators, then checking whether a given map is an endomorphism of G amounts to checking whether every defining relator is taken by this map to an element equal to 1 in G . Thus, the word problem in G has to be efficiently solvable.

Checking whether a given map is an automorphism is more complex, and there is no general recipe for doing that, although for a particular platform group that we describe in Section 8.5.7 this can be done very efficiently. In general, it would make sense for Alice to supply a proof (at the response step) that her ψ is an automorphism; this proof would then depend on an algorithm Alice used to produce ψ .

PROPOSITION 8.5.7. *Suppose that after several runs of steps (2)-(3) of the above protocol, both values of c are encountered. Then successful forgery is equivalent to finding an endomorphism φ such that $\varphi(g) = h$, and is therefore NP-hard in some groups (and rings) G .*

The proof is similar to that of Proposition 8.5.6. We also note that in [112], a class of rings is designed for which the problem of existence of an endomorphism between two given rings from this class is NP-hard.

A particular example of a group with the NP-hard endomorphism problem is given in the following subsection.

8.5.7. Platform: free metabelian group of rank 2. A group G is called *abelian* (or commutative) if $[a, b] = 1$ for any $a, b \in G$, where $[a, b]$ is the notation for $a^{-1}b^{-1}ab$. This can be generalized in different ways. A group G is called *metabelian* if $[[x, y], [z, t]] = 1$ for any $x, y, z, t \in G$. The commutator subgroup of G is the group $G' = [G, G]$ generated by all commutators, i.e., by expressions of the form $[u, v] = u^{-1}v^{-1}uv$, where $u, v \in G$. The second commutator subgroup G'' is the commutator of the commutator of G .

DEFINITION 8.5.8. Let F_n be the free group of rank n . The factor group F_n/F_n'' is called the *free metabelian group* of rank n , which we denote by M_n .

Roman'kov [236] showed that, given any Diophantine equation E , one can efficiently (in linear time in the “length” of E) construct a pair of elements u, v of the group M_2 , such that to any solution of the equation E , there corresponds an endomorphism of M_2 that takes u to v , and vice versa. Therefore, there are pairs of elements of M_2 for which the endomorphism problem is NP-hard (see e.g. [89, Problem AN8]). Thus, if a free metabelian group is used as the platform for the protocol in this section, then, by Proposition 8.5.7, forgery in that protocol is NP-hard.

8.5.8. Platform: \mathbb{Z}_p^* . Here the platform group is \mathbb{Z}_p^* , for a prime p . Then, since \mathbb{Z}_{p-1}^* acts on \mathbb{Z}_p^* by automorphisms, via the exponentiation, this can be used as the platform for the Protocol II. In this case, forgery is equivalent to solving the discrete logarithm problem, by Proposition 8.5.7.

8.6. No-leak authentication by the Sherlock Holmes method

Here we follow the exposition in [114]. In this section, we propose a class of authentication schemes that are *literally* zero-knowledge, as compared to what is formally defined as “zero-knowledge” in cryptographic literature. We call this “no-leak” authentication to distinguish from an established “zero-knowledge” concept. The “no-leak” condition implies “zero-knowledge” (even “perfect zero-knowledge”), but it is actually stronger, as we illustrate by examples.

The principal idea behind our schemes is: the verifier challenges the prover with questions that he (the verifier) already knows answers to; therefore, a verifier

who follows the protocol cannot *possibly* learn anything new during any number of authentication sessions. This is therefore also true for a passive adversary, even computationally unbounded one.

The schemes proposed in this section can also be used for encryption.

8.6.1. No-leak vs. zero-knowledge. We call our proposal a “no-leak” authentication because the “zero-knowledge” terminology is “trademarked” in cryptographic literature by way of specific formal definitions; see e.g. [99]. Unfortunately, “zero-knowledge” (or even “perfect zero-knowledge”) is just a euphemism, i.e., those formal definitions do not actually imply that there is no leak of information during any proof session. Even more unfortunately, it seems to be a rather common misconception these days that “perfect zero-knowledge” implies no leak for *any* honest verifier; this is what often happens with euphemisms. We refer the interested reader to [218] where an “hierarchy of zero-knowledge” is suggested, so that the presence of leak or lack thereof depends on the (honest) verifier’s computational abilities.

Here we compare the established definition(s) of “perfect zero-knowledge” to our definition of “no-leak” that applies, incidentally, to a computationally unbounded forger as well. Perhaps the difference is best illustrated by scrutinizing the Graph ISO protocol from [101], which is known to be “perfect zero-knowledge”, but it is obviously *not* “no-leak”. We discuss this protocol in detail in our Section 8.6.5, while now we will try to explain the difference in definitions of “perfect zero-knowledge” vs. “no-leak”. We note that there are several somewhat different definitions of “perfect zero-knowledge” (for example, the seminal paper [101] has 3 different definitions), so here we are just trying to capture most essential features of those definitions to help us make our point.

DEFINITION 8.6.1 (“Perfect zero-knowledge”). Let (P, V) be an interactive proof system for a language L . Let F be a polynomial-time probabilistic Turing machine (PTM) that produces forged transcripts. Let $\mathcal{T}(x)$ denote the set of all possible true transcripts (obtained by the verifier V actually engaging in an interactive proof with the prover P on input x), and $\mathcal{F}(x)$ the set of all possible forged transcripts. The proof system (P, V) is called *perfect zero-knowledge* (for L) if there exists a forger F such that for any $x \in L$:

- (i) the set of forged transcripts is identical to the set of true transcripts, i.e., $\mathcal{F}(x) = \mathcal{T}(x)$;
- (ii) the two associated probability distributions are identical, i.e., for any transcript $T \in \mathcal{T}(x)$, one has $\Pr_{\mathcal{T}}[T] = \Pr_{\mathcal{F}}[T]$.

To better illustrate the difference between this and the following definition, we say, somewhat informally, that the former implies that in a perfect zero-knowledge proof system, if a specific leak of information (about the prover’s secret key, say) can occur with probability p during an interaction between V and P , then it can occur with the same probability p if V does not interact with P at all, but just simulates such interaction.

By comparison, in the following definition the condition is (again, informally) that no leak should occur at all, i.e, the probability p alluded to in the previous paragraph is always 0. The following definition is an attempt to formalize this idea. Note also that in our definition, the “polynomial-time” restriction on the forger is dropped.

DEFINITION 8.6.2 (“No-leak”). Let (P, V) be an interactive proof system for a language L . It is called *no-leak* if, for any $x \in L$ and for any function f from L to \mathbb{N} , any sequence $\mathcal{T}(x)$ of true transcripts obtained in time $\leq f(x)$ by interaction of the verifier V with the prover P on input x , could be also constructed by a deterministic algorithm by V alone (i.e., without interacting with P) in time $\leq f(x)$.

In other words, the prover can contribute nothing whatsoever to the verifier’s ability of producing any sequence of true transcripts.

Probably the easiest way to achieve the “no-leak” property is to have the verifier V only ask those questions that he already knows the correct answers to. In this scenario, the corresponding proof system will obviously be perfect zero-knowledge since whatever V can compute after interacting with the prover, he could already have computed *before* interacting with the prover. At the same time, “perfect zero-knowledge” does not necessarily imply “no leak”; as we have already mentioned, this is best illustrated by scrutinizing the graph ISO protocol from [101], so the reader who is not convinced at this point that the “no-leak” condition is stronger than “perfect zero-knowledge”, is referred to our Section 8.6.5.

8.6.2. The meta-protocol for authentication. It is well known that the main motivation for studying zero-knowledge proof systems is their application to authentication, which is also our main motivation here. The reason why we call our idea of authentication the “Sherlock Holmes method” is the following. It is the case with most natural decision problems in algebra (such as the identity problem, the conjugacy problem, the membership problem, etc.) that, for a generic input, the “no” answer can be obtained much more efficiently than the “yes” answer. Thus, if the prover “eliminates the impossible” by (efficiently) getting the “no” answers whenever she can, she is left with the only remaining possibility for a “yes” answer. We note that in a typical concrete realization of this idea, the prover will not be able to give a “yes” answer by any other method than “eliminating the impossible”, which is why we call it the “Sherlock Holmes method.”

To conclude the introductory remarks, we summarize what we think are the most interesting features of our proposal:

- (1) A verifier who follows the protocol cannot possibly obtain from the prover any information that he does not already know. This is therefore also true for a passive adversary, even computationally unbounded one.
- (2) There is no “concrete” problem for the adversary to solve in order to obtain the prover’s long-term private key from her public key. The problem he/she faces is to obtain a test for non-membership in a set that he/she does not know.

Now a natural question is whether a verifier who does *not* follow the protocol can obtain any information during an authentication session by offering challenges without knowing *a priori* what the correct response should be. We comment on this question at the end of this subsection.

Finally, we note that our general scheme can also be used for encryption, see Remark 8.6.3 at the end of this subsection.

Now we give a description of our general idea of “authentication by the Sherlock Holmes method”, leaving particular realizations to the next sections. Here Alice is the prover and Bob the verifier.

Alice's private key consists of: (a) a subset S_0 of some "universal" set S ; (b) an efficient test telling that a given element of S does *not* belong to S_0 ; (c) a way to disguise S_0 to some S'_0 , e.g., a self-bijection φ of the universal set S , such that $\varphi(S_0) = S'_0$ and it is possible for Alice to efficiently compute both φ and φ^{-1} . In some realizations, (c) is not really necessary, and the role of S'_0 is played just by a subset of S_0 .

Alice's public key is a pair of sets S'_0, S_1 that either are disjoint or have negligible intersection. *Note that the verifier does not know whether Alice has a "non-membership test" for S'_0 or for S_1 ; this information is not public!* Both sets are given to the public in such a way that it is possible to efficiently select ("sample") a random element from either set.

We note that the idea of a "non-membership test" for S_0 (see part (b) of Alice's private key above) can be expressed in a more precise language. Alice should have her private *separator* T , such that $S_0 \subset T$, the intersection $T \cap S_1$ is empty or negligible, and T is a "nice" set in the sense that the problem of membership in T is efficiently solvable. Thus, Alice can efficiently check membership of an element x in question in the set T ; then, if $x \notin T$, she knows for sure that $x \notin S_0$. If $x \in T$, then she *assumes* that $x \in S_0$; the "tighter" (i.e., the closer to S_0) T is, the more chances this assumption has to be correct. Thus, even though there might be many different separators for a given pair of sets, a good separator is hard for the adversary to find without knowing the set S_0 .

The protocol itself is the following sequence of steps.

- (1) Bob selects a tuple (x'_1, \dots, x'_{2m}) of random elements from either S'_0 or S_1 , with exactly m elements from S'_0 and exactly m elements from S_1 , and sends it to Alice.
- (2) Alice checks, using her private test, for every i , whether for the element x_i corresponding to x'_i , one has $x_i \notin S_0$. If her test fails, Alice assumes that $x_i \in S_0$ (equivalently, $x'_i \in S'_0$). Then she sends a tuple of $\frac{m}{2}$ "1"s to Bob, in $\frac{m}{2}$ randomly selected places corresponding to $x'_i \notin S'_0$. She gives no indication of the results of her test for the remaining $\frac{3m}{2}$ places.
- (3) Bob, who knows the right answer, simply compares it to Alice's response and accepts or rejects authentication accordingly.

To prevent the adversary from guessing the right answer with non-negligible probability, several rounds of this protocol may be run (depending on what probability is accepted as "negligible"); this is similar to the Feige-Fiat-Shamir scheme [74]. We note that if, say, $m = 20$, then the probability of guessing the right answer in a single session is less than 10^{-6} .

To conclude this subsection, we make the following remark.

REMARK 8.6.3. The protocol in this section can also be used for encryption. Namely, if Bob wants to transmit an encrypted bit to Alice, he sends her a random element from S'_0 in case he wants to transmit a "0", and a random element from S_1 in case he wants to transmit a "1".

We do not pursue this direction here, and in particular, we do not make any claims concerning security of relevant encryption schemes.

8.6.3. Correctness of the protocol. The protocol is perfectly correct in the sense that if Alice is in possession of her private key and both parties follow all steps of the protocol, then Alice will be able to answer all Bob's challenges correctly with

probability 1, and Bob will then accept Alice’s authentication with probability 1. Even if the intersection $T \cap S_1$ is not empty and Alice’s test fails for some elements sent by Bob, this will not affect Alice’s response because her response only includes places where her test did not fail.

Also, since Bob does not obtain from Alice any information that he does not already know, he can construct by a deterministic algorithm any true authentication session transcript, without interacting with Alice. Thus, the following is obvious:

PROPOSITION 8.6.4. *No information about the prover’s private key is leaked during any authentication session with an honest verifier in the above authentication scheme, i.e., it is a “no-leak” proof system in the sense of Definition 8.6.2.*

In Sections 8.6.6 and 8.6.7, we give two particular realizations of our meta-protocol just to illustrate the diversity of possible applications of our main idea. We emphasize once again though that we avoid detailed technical discussions (in particular, making quantitative statements) here and keep the focus on theoretical aspects.

8.6.4. Questions and answers. In this section, we answer some questions and concerns that a reader may have at this point; these questions have actually been asked by some of our colleagues. We thought that arranging this material in the “FAQ format” *after* describing our meta-protocol would be more helpful to the reader than if we made it part of the introductory remarks.

Q. Do you prove that recovering the prover’s private key from her public key in any of your particular realizations is computationally infeasible? In particular, do you prove the existence of one-way functions?

A. No and no. Moreover, our focus in this proposal is on leak (or lack thereof) of information during authentication sessions, rather than on security of the prover’s public key construction.

Q. If not, then what is the novelty of your proposal?

A. The main novelty is that in our authentication scheme, a verifier who follows the protocol cannot possibly obtain from the prover any information that he does not already know. This is also true for a passive adversary, even computationally unbounded one. (Of course, a computationally unbounded adversary can usually find correct responses to the verifier’s challenges by using “brute force”, but this is a different story.)

Q. In that case, what is the difference between your protocol and the classical zero-knowledge Graph Non-ISO protocol [103] where the prover convinces the verifier that two given graphs are non-isomorphic by giving to the verifier only the information that he already knows?

A. In the Graph Non-ISO protocol [103] the prover is assumed to be computationally unbounded (i.e., there is no “trapdoor”). Therefore, the Graph Non-ISO protocol cannot be used for authentication purposes in any meaningful scenario in real life.

Q. Well then, how about well-known zero-knowledge protocols with trapdoor, such as Graph ISO, or quadratic residuosity, or Feige-Fiat-Shamir scheme?

A. None of these protocols is “no-leak”; we think it is best illustrated by analyzing the Graph ISO protocol; this is what we do in Section 8.6.5 below.

Q. What about a verifier who does *not* follow the protocol? What if he just produces some random challenges? Can he obtain some information about the prover’s private key from her responses in that case?

A. The answer to this question may depend on a particular realization of our meta-protocol; more specifically, on the size of the sets S'_0 and S_1 relative to the size of the “universal” set S from which a “frivolous” verifier can pick up his random challenges. Typically, the sets S'_0 and S_1 are negligible in S , which implies that if a frivolous verifier picks his challenges randomly from S , he will get the “not in S_0 ” response for every random element from S . Whether or not this can give him any information about S_0 other than S_0 is negligible in S , is not entirely clear, although the common sense suggests that it cannot. We do not make any claims to that effect here.

8.6.5. Why is the Graph ISO proof system not “no-leak”? In this section, we try to resolve the confusion that stems from taking the “perfect zero-knowledge” euphemism too literally. To that end, we recall the well-known Graph ISO protocol from [101]. Here Alice is the prover and Bob the verifier.

- (1) Alice’s public key consists of two isomorphic graphs, Γ_0 and Γ_1 . Alice’s private key is an isomorphism $\varphi : \Gamma_1 \rightarrow \Gamma_0$. Alice is supposed to prove to Bob that Γ_0 is isomorphic to Γ_1 .
- (2) To begin a session, Alice selects a random bit b , a random isomorphism $\sigma : \Gamma_b \rightarrow H$, and sends the “commitment” graph H to Bob.
- (3) Bob chooses a random bit c and sends it to Alice.
- (4) Alice responds with an isomorphism $\tau : \Gamma_c \rightarrow H$.
- (5) Bob verifies that τ is, indeed, an isomorphism.

Obviously, if the same graph H appears as commitment in two different sessions, and if the corresponding challenge bits c are different, then Bob, who gets isomorphisms $\tau_0 : \Gamma_0 \rightarrow H$ and $\tau_1 : \Gamma_1 \rightarrow H$, can recover the isomorphism $\tau_0^{-1}\tau_1$ between Γ_1 and Γ_0 . Therefore, after two sessions, Bob can obtain, with non-zero probability, information that he did not have before he started to interact with Alice. Thus, it is easy to see that the above proof system is not “no-leak” in the sense of Definition 8.6.2. Indeed, if interaction between Bob Alice produces a sequence of, say, just two transcripts with the same commitment graph H and different corresponding challenge bits c , then to reproduce this sequence with probability 1, Bob would need a lot more than just 2 attempts.

At the same time, this proof system is “perfect zero-knowledge” in the sense of Definition 8.6.1 because Bob can obtain the same information *with the same probability* (but not with probability 1!) if he simulates the protocol by himself, without interacting with Alice. Specifically, by selecting random isomorphisms $\alpha : \Gamma_0 \rightarrow H_1$ and $\beta : \Gamma_1 \rightarrow H_2$, he may end up, with non-zero probability, with $H_1 = H_2$, hence recovering an isomorphism between Γ_1 and Γ_0 .

8.6.6. A particular realization: subset sum. In this section, we offer a particular realization of the meta-protocol from Section 8.6.2, exploiting the hardness of the subset sum problem; see e.g. [89]. We note that the complexity of this particular problem was previously used in [139] in different cryptographic contexts,

namely for constructing a pseudo-random generator and a universal one-way hash function.

The “universal” set S in this section is the set of all m -tuples of m -dimensional vectors over \mathbf{Q} .

Alice’s private key is a set $S_0 = \{a_1, \dots, a_m\}$ of m random linearly independent (over \mathbf{Q}) m -dimensional vectors *with integer coordinates*, which is therefore a basis of \mathbf{Q}^m . However, the vector a_1 is special: the g.c.d. of its coordinates is even.

Alice’s public key includes the vector a_1 and a set of $k > m$ random vectors c_1, \dots, c_k from the \mathbf{Z}_+ -span of S_0 .

Now we give a description of the authentication protocol. *To simplify the notation, we give an exposition where Bob challenges Alice with just a single element rather than with a tuple of elements, as in the meta-protocol in Section 8.6.2.*

- (1) Bob selects, with equal probabilities, either a random vector $c \in \text{Span}_{\mathbf{Z}_+}(a_1, c_1, \dots, c_k)$ or a random vector $c \in \text{Span}_{\mathbf{Z}_+}(\frac{1}{2}a_1, c_1, \dots, c_k)$ and sends the vector c to Alice; in the latter case, Bob takes care that the coefficient at $\frac{1}{2}a_1$ is odd. Here $\text{Span}_{\mathbf{Z}_+}$ denotes the set of all linear combinations of given vectors *with nonnegative integer coefficients*.
- (2) Alice, using standard linear algebra, finds (rational) coordinates of c in the basis S_0 . If at least one of these coordinates is not a nonnegative integer, she knows that $c \notin \text{Span}_{\mathbf{Z}_+}(a_1, c_1, \dots, c_k)$; therefore, she sends “1” to Bob. If all coordinates are nonnegative integers, Alice assumes that $c \in \text{Span}_{\mathbf{Z}_+}(a_1, c_1, \dots, c_k)$, and sends “0” to Bob.
- (3) Bob, who knows the right answer, simply compares it to Alice’s response and accepts or rejects authentication accordingly.

We note that there is a negligible probability for Bob to reject a legitimate Alice because it may happen that all coordinates of c in the basis S_0 are nonnegative integers, but $c \notin \text{Span}_{\mathbf{Z}_+}(a_1, c_1, \dots, c_k)$. It may, in fact, even happen (again, with negligible probability) that $c \in \text{Span}_{\mathbf{Z}_+}(a_1, c_1, \dots, c_k)$, but Bob expected Alice to respond with a “1” because he selected his $c \in \text{Span}_{\mathbf{Z}_+}(\frac{1}{2}a_1, c_1, \dots, c_k)$.

We also note that the reason for using a public vector a_1 with g.c.d. of coordinates even is to have Bob’s vector c in $\text{Span}_{\mathbf{Q}_+}(a_1, c_1, \dots, c_k)$ in either case, because there is a polynomial-time test detecting whether or not a given vector belongs to the \mathbf{Q}_+ -span of other given vectors (cf. linear programming problem); see [155] or [244].

Finally, we note that the problem that the adversary who wants to impersonate the prover faces is the following: find out whether or not the matrix equation $Bx = c$ has a solution for x as a vector with nonnegative integer coordinates. Here B is the matrix made up of coordinates of the vectors a_1, c_1, \dots, c_k , c is the challenge vector selected by Bob, and x is the vector unknown to both the prover and the adversary. A special case of this problem, where B is just a vector with integer coordinates, x is a 0-1 vector, and c is just an integer, is known as the *subset sum problem* and is NP-complete; see e.g. [89]. Moreover, as pointed out, for example, in [99, p. 41], it appears that the subset sum problem might be hard on random instances, not just on some carefully selected ones.

Suggested parameters and key generation. Suggested parameter values for the protocol above are:

- (1) The dimension of vectors is $m = 20$.

- (2) Coordinates of the vectors a_i : random nonnegative integers ≤ 10 . We note that m random m -dimensional vectors like that are going to be linearly independent with overwhelming probability.
- (3) Vectors c_i are constructed by Alice as random linear combinations of the vectors a_i with nonnegative integer coefficients ≤ 10 . The number of vectors c_i is $k = 2m$.
- (4) At step (1) of the protocol, Bob constructs his vector c as a random linear combination of the public vectors a_1, c_1, \dots, c_k with nonnegative integer coefficients ≤ 10 , with one possible exception: according to the protocol description, he may choose the coefficient at a_1 to be of the form $\frac{n}{2}$, where n is odd, $1 \leq n \leq 19$.

8.6.7. A particular realization: polynomial equations. In this section, we offer another particular realization of the meta-protocol from Section 8.6.2.

Alice's private key consists of: (i) a polynomial $h(x_1, \dots, x_k)$ over \mathbf{Z} ; (ii) a large prime p ; (iii) a constant $c \in \mathbf{Z}$.

Alice's public key includes: (i) polynomial $f(x_1, \dots, x_k) = (h(x_1, \dots, x_k))^2 - c \pmod{p}$. Thus, for any $x_1, \dots, x_k \in \mathbf{Z}$, there is $u \in \mathbf{Z}$ such that $f(x_1, \dots, x_k) + c = u^2 \pmod{p}$; (ii) a random polynomial $g(x_1, \dots, x_k)$ over \mathbf{Z} with the same collection of monomials as in f and with coefficients of the same magnitude as in f .

Now we give a description of the authentication protocol. Again, *to simplify the notation, we give an exposition where Bob challenges Alice with just a single element rather than with a tuple of elements, as in the meta-protocol in Section 8.6.2.*

- (1) Bob selects random integers x_1, \dots, x_k and plugs them, with equal probabilities, into either f or g . He then sends the result, call it $\text{Bob}(x_1, \dots, x_k)$, to Alice.
- (2) Alice computes $a = \text{Bob}(x_1, \dots, x_k) + c \pmod{p}$ and checks whether or not a is a square modulo p . If not, she knows that $\text{Bob}(x_1, \dots, x_k) \neq f(x_1, \dots, x_k)$ and sends "1" to Bob. If it is, Alice assumes that $\text{Bob}(x_1, \dots, x_k) = f(x_1, \dots, x_k)$ and sends "0" to Bob.
- (3) Bob, who knows the right answer, simply compares it to Alice's response and accepts or rejects authentication accordingly.

The way Alice checks whether or not a is a square modulo p is as follows. She raises a to the power of $\frac{p-1}{2}$. If the result is equal to 1 modulo p , then a is a square modulo p ; if not, then not.

Again, we note that there is a negligible probability for Bob to reject a legitimate Alice because it may happen that $\text{Bob}(x_1, \dots, x_k) + c$ is a square modulo p , but $\text{Bob}(x_1, \dots, x_k) = g(x)$.

Suggested parameters and key generation. Suggested parameter values for the protocol above are:

- (1) The number k of variables: between 3 and 5.
- (2) The value of p : on the order of 2^t , where t is the security parameter.
- (3) The degree of Alice's private polynomial h : between 2 and 3. The magnitude of its coefficients: at least $\frac{p}{2}$.
- (4) Bob generates his integers x_1, \dots, x_k uniformly randomly from the interval $[1, 2^{\frac{t}{k}}]$.

REMARK 8.6.5. The adversary may try to attack Bob’s challenge by solving one of the equations $f(x_1, \dots, x_k) = \text{Bob}(x_1, \dots, x_k)$ or $g(x_1, \dots, x_k) = \text{Bob}(x_1, \dots, x_k)$ for integers x_1, \dots, x_k , or just try to find out whether either of these equations has integer solutions. The corresponding decision problem (the Diophantine problem, or Hilbert’s 10th problem) is known to be undecidable; see [186]. In our situation, however, adversary actually faces a *promise problem* since he/she knows that at least one of the equations has integer solutions. Furthermore, in our situation the range for the unknowns is bounded. Still, the “bounded” Diophantine problem is known to be NP-hard; see e.g. [89], which makes this kind of attack look infeasible.

Part 3

Generic Complexity and Cryptanalysis

In this part of the book, we discuss two measures of complexity of an algorithm, average-case and generic-case, and argue that it is the generic-case complexity (we often call it just generic complexity) that should be considered in the context of cryptanalysis of various cryptographic schemes.

Since in our book we mostly deal with cryptographic schemes based on non-commutative infinite groups, we give a survey of what is known about generic-case behavior of various algorithms typically studied in combinatorial group theory and relevant to cryptography. The very idea of “genericity” in group theory was introduced by Gromov and Ol’shanskii and is now a subject of very active research. Genericity exhibits itself on many different levels in algebraic and algorithmic properties of “random” algebraic objects and in the generic-case behavior of their natural geometric invariants. A generic approach often leads to the discovery of objects with genuinely new and interesting algebraic properties. For example, genericity provides a totally new source of *group-theoretic rigidity*, quite different from the standard source provided by lattices in semisimple Lie groups. Generic-case analysis of group-theoretic algorithms gives a much more detailed and stratified “anatomical” picture of an algorithm than worst-case analysis provides. It also gives a better measure of the overall “practicality” of an algorithm and often leads to genuine *average-case* results.

It is probably safe to say that by now it is pretty clear that the worst-case complexity is not necessarily a good measure of “practicality” of an algorithm. The paradigm example is Dantzig’s Simplex Algorithm for linear programming. Very clever examples of Klee and Minty [159] show that the Simplex Algorithm can be made to take exponential time. However, the simplex method runs thousands of times a day in many real-life applications and it always works very fast, namely in linear time. Although there are provably polynomial-time algorithms for linear programming, these have not replaced the Simplex Algorithm in practice. It turns out that this type of phenomenon, which we refer to as having low “generic-case complexity” is very pronounced in group theory. This notion was introduced and formalized by Kapovich, Myasnikov, Schupp and Shpilrain in [146]. In particular, they showed that for most groups usually studied in combinatorial or geometric group theory, the generic-case complexity of the word problem is linear time, often in a very strong sense. Consequently, the actual average-case complexity is also often linear time.

We note that unlike the average-case complexity, generic-case complexity completely disregards potentially bad behavior of an algorithm on an asymptotically negligible set of inputs. Thus generic-case complexity of an algorithm \mathcal{A} may be low (e.g. linear time) even if the relevant algorithmic problem is undecidable (and \mathcal{A} has infinite running time on some inputs), while the average-case complexity would necessarily be infinite in this case. As was observed in [146], this is exactly what happens with the classical decision problems in group theory, such as the word, conjugacy and subgroup membership problems. Moreover, in [147] the same authors were also able to apply their generic-case methods to obtain genuine average-case complexity conclusions. The basic idea there is that by running in parallel a total algorithm with subexponential worst-case complexity and a partial algorithm with low *strong generic-case* complexity, one obtains an algorithm with low average-case complexity. Typical results of [147] imply, for example, that for all braid groups,

all groups of knots with hyperbolic complements and all Artin groups of extra large type, the average-case complexity of the word problem is linear time.

Apart from shedding light on “practicality” of various algorithms, generic-case analysis has some additional benefits. First, it provides a natural stratification of the set of inputs of a problem into the “easy” and “not necessarily easy” parts. Concentration on the latter part often forces one to redefine the original problem and substantially change its nature. By iterating this process one can obtain a detailed analysis of the “anatomy” of an algorithmic problem that is in many ways more informative than the worst-case analysis. Isolating the “hard” part of an algorithmic problem may be important for possible practical applications, in particular, to cryptanalysis of various cryptographic schemes. This is what we explore in more detail in Part 4 of this book.

Moreover, as our experience shows, generic-case considerations can lead to the discovery of new algorithms with better average-case and worst-case behavior.

Finally, we should emphasize that one must not confuse the fact that generic results are often easy to state with simplicity of their proof. Proofs usually require interaction of some aspects of probability theory with deep results about algebraic questions under consideration.

CHAPTER 9

Distributional Problems and the Average Case Complexity

9.1. Distributional computational problems

One of the aims of the section is to set up a framework that would allow one to analyze behavior of algorithms at large, to study expected or average properties of algorithms, or its behavior on “most” inputs. The key point here is to equip algorithmic problems with probability distributions on their sets of instances.

9.1.1. Distributions and computational problems. To study behavior of algorithms on average or on “most” inputs one needs to have a distribution on the set of inputs. In this section we discuss typical distributions that occur on discrete sets of inputs.

DEFINITION 9.1.1. A *distributional computational problem* is a pair (\mathcal{D}, μ) where $\mathcal{D} = (L, I)$ is a computational problem and μ is a probability measure on I .

The choice of μ is very important, it should be natural in the context of the problem. Quite often, the set I is infinite and discrete (enumerable), so, unlike in the finite set case, it is impossible to use uniform distributions on I . Discreteness of the set I usually implies that the probability distributions μ are *atomic*, i.e., $\mu(x)$ is defined for every singleton $\{x\}$ and for a subset $S \subseteq I$

$$(10) \quad \mu(S) = \sum_{x \in S} \mu(x).$$

It follows that an atomic measure μ is completely determined by its values on singletons $\{x\}, x \in I$, so to define μ it suffices to define a function $p : I \rightarrow \mathbb{R}$ (with $\mu(\{x\}) = p(x)$), which is called a *probability mass function* or a *density function* on I , such that $p(x) \geq 0$ for all $x \in I$, and $\sum_{x \in I} p(x) = 1$.

There are two general basic ideas on how to introduce a natural distribution on I .

- (1) *Distributions via generating procedures.* If elements of I can be naturally generated by some “random generator” R then probability $\mu(x)$ can be defined as the probability of the generator R to produce x . We refer to [196] for a detailed discussion.
- (2) *Distributions via stratifications.* Let $s : I \rightarrow \mathbb{N}$ be a size function on I . An atomic distribution μ on I respects the size function s if:

$$\forall x, y \in I (s(x) = s(y) \longrightarrow \mu(x) = \mu(y)).$$

Such a measure μ on I is termed *size invariant* or *homogeneous* or *uniform*. More about homogeneous measures can be found in [13]. Here we give only some facts that are required in the sequel.

Observe, that a homogeneous distribution μ induces a uniform distribution on each sphere I_k , so if the induced distribution is non-zero then the sphere is finite. The following result describes size-invariant distributions on I .

LEMMA 9.1.2. *Let μ be an atomic measure on I and s a complexity function on I with finite spheres I_k . Then:*

- (1) *if μ is s -invariant, then the function $d_\mu : \mathbb{N} \rightarrow \mathbb{R}$ defined by*

$$d_\mu : k \rightarrow \mu(I_k)$$

is an atomic probability measure on \mathbb{N} ;

- (2) *if $d : \mathbb{N} \rightarrow \mathbb{R}$ is an atomic probability measure on \mathbb{N} , then the function $p_{s,d} : I \rightarrow \mathbb{R}$ defined by*

$$p_{s,d}(x) = \frac{d(s(x))}{|I_n|}$$

is a probability density function which gives rise to an atomic s -invariant measure on I .

Proof is obvious. □

REMARK 9.1.3. Since $d(k) \rightarrow 0$ as $k \rightarrow \infty$, the more complex (with respect to a given size function s) elements of a set $R \subset I$ contribute less to $\mu_{s,d}(R)$, so there is a bias to elements of small size.

This discussion shows that the homogeneous probability measures $\mu_{s,d}$ on I are completely determined by the complexity function $s : I \rightarrow \mathcal{N}$ and a moderating distribution $d : \mathbb{N} \rightarrow \mathbb{R}$. Here are some well-established parametric families of density functions on \mathbb{N} that will occur later on in this framework:

The *exponential density*:

$$d_\lambda(k) = (1 - e^{-\lambda})e^{-\lambda k};$$

The *Cauchy density*:

$$d_\lambda(k) = b \cdot \frac{1}{(k - \lambda)^2 + 1};$$

The *Dirac density*:

$$d_m(k) = \begin{cases} 1 & \text{if } k = m, \\ 0 & \text{if } k \neq m. \end{cases}$$

The following density function depends on a given size function s on I :

The *finite disc uniform density*:

$$d_m(k) = \begin{cases} \frac{|I_k|}{|B_m(I)|} & \text{if } k \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

For example, the Cauchy density functions are very convenient for defining degrees of polynomial growth “on average”, while the exponential density functions are suitable for defining degrees of exponential growth “on average”. Exponential density functions appear naturally as distributions via random walks on graphs or as *multiplicative* or *Boltzmann* distributions (see [29] for details).

Other distributions arise from different random generators of elements in I . For example, Dirac densities correspond to the uniform random generators on the spheres I_n ; finite disc densities arise from uniform random generators on the discs $B_n(I)$.

9.1.2. Stratified problems with ensembles of distributions. In this section we discuss stratified computational problems \mathcal{D} with ensembles of distributions. In this case the set of instances I of the problem \mathcal{D} comes equipped with a collection of measures $\{\mu_n\}$ each of which is defined on the sphere I_n (or a ball B_n). In this case we do not assume in advance that there exists a measure on the set I .

Let \mathcal{D} be a stratified computational problem with a set of instances $I = I_{\mathcal{D}}$ and a size function $s = s_{\mathcal{D}} : I \rightarrow \mathbb{R}$. If for every n the sphere I_n comes equipped with a probability distribution μ_n then the collection of measures $\mu = \{\mu_n\}$ is called a *spherical ensemble* of distributions on I . Similarly, a *volume ensemble* of distributions $\mu = \{\mu_n\}$ is a collection of distributions such that μ_n is a distribution on the ball B_n and such that μ_{n-1} is the measure induced from μ_n on B_{n-1} . Here, and in all similar situations, we extend the standard definition of the induced measure μ_S to subsets $S \subseteq B_n$ of measure zero ($\mu_n(S) = 0$) defining $\mu_S(w) = 0$ for all $w \in S$.

EXAMPLE 9.1.4. Suppose the spheres I_n are finite for every $n \in \mathbb{N}$. Then the uniform distribution μ_n on I_n gives a spherical ensemble of distributions $\mu = \{\mu_n\}$ on I .

The following is a typical way to define ensembles of distributions on I . We say that a probability distribution μ on I is *compatible* with the size function $s : I \rightarrow \mathbb{N}$ if s is μ -measurable, i.e., for every n the sphere I_n is a μ -measurable set. Now, if μ is a probability distribution on I compatible with s then for every n such that $\mu(I_n) \neq 0$ μ induces a measure μ_n on I_n . We extend this definition for all n defining $\mu_n(w) = 0$ for every $w \in I_n$ in the case $\mu(I_n) = 0$.

This gives an induced ensemble of spherical distributions $\{\mu_n\}$ on I . For instance, the probability distributions defined in Section 9.1.1 give rise to typical spherical ensembles of distributions on I .

In some sense the converse also holds. If $\{\mu_n\}$ is a spherical ensemble of distributions on I then one can introduce a distribution $\tilde{\mu}_d$ on I which induces the ensemble μ . Indeed, let $d : \mathbb{N} \rightarrow \mathbb{R}$ be a moderating distribution on \mathbb{N} , so $\sum_n d(n) = 1$. For a subset $R \subseteq I$ such that $R_n = R \cap I_n$ is μ_n -measurable for every n define $\tilde{\mu}_d(R)$ as

$$\tilde{\mu}_d(R) = \sum_n d(n) \mu_n(R_n).$$

Clearly, the series above converges since $\sum_n d(n) \mu_n(R_n) \leq \sum_n d(n) = 1$. It is easy to see that $\tilde{\mu}_d$ is a distribution on I that induces the initial ensemble μ on the spheres of I . Notice, that if μ_n is an atomic distribution on I_n then μ_d is an atomic distribution on I .

The argument above shows that under some natural conditions one can harmlessly switch between distributions on I and the spherical ensembles of distributions on I . Similar constructions hold for volume ensembles as well.

9.1.3. Randomized many-to-one reductions. Very important theoretical class of reductions is many-to-one reductions done by a *probabilistic Turing machine* called *randomized many-to-one reductions*. Intuitively a probabilistic Turing machine is a Turing machine with a random number generator. More precisely it is a machine with 2 transition functions δ_1 and δ_2 . At each step of computations with probability 50% the function δ_1 is used and with probability 50% the function δ_2 is used. So, probabilistic Turing machine is the same as nondeterministic Turing machine with the only difference being how we interpret its computations.

If for NTM M we are interested if there exists a sequence of choices that make M accept a certain input, for the same PTM M we are interested in which probability acceptance happens.

DEFINITION 9.1.5. Let D be a decision problem. We say that a PTM M decides D if it outputs that right answer with probability at least $2/3$.

DEFINITION 9.1.6. We say that a PTM M is *polynomial time on average* if there exists $c > 0$ such that

$$\sum_{x,z} T_M^c(x, z) \mu'(x) P(z) < \infty,$$

i.e., complexity function $T_M^c(x, z)$ is polynomial time on average taken over the distribution μ_1 on inputs x and the distribution P on internal coin tosses z .

DEFINITION 9.1.7. Let D_1 and D_2 be decision problems. We say that a probabilistic Turing machine M *randomly reduces* D_1 to D_2 if for any x ,

$$P(z \mid x \in D_1 \Leftrightarrow M(x, z) \in D_2) \geq 2/3,$$

where probability P is taken over all internal coin tosses z .

9.2. Average case complexity

In this section we briefly discuss some principle notions of the average-case complexity. There are several different approaches to the average-case complexity, but in some sense, they all involve computing the expected value of the running time of an algorithm with respect to some measure on the set of inputs.

More generally, one can develop a theory of average case behavior of arbitrary functions $f : I \rightarrow \mathbb{R}^+$, not only the time functions of algorithms, provided the set I is a probability space equipped with a size function. From this standpoint, the average case complexity of algorithms appears as a particular application of the average case analysis to the time (or space, or any other resource) functions of algorithms.

Below we assume, if not said otherwise, that I is a set (of “inputs”) with a size function $s : I \rightarrow \mathbb{N}$ and a probability measure μ . Throughout this section we assume also that the set I is *discrete* (finite or countable) and the measure μ on I is *atomic*, so all functions defined on I are μ -measurable. Notice, however, that all the results and definitions hold in the general case as well, if the size function s and all the functions in the consideration are μ -measurable.

Sometimes, instead of the measure μ we consider ensembles of spherical (or volume) distributions $\{\mu_n\}$ on spheres I_n (balls B_n) of I relative to the size function s . This approach seems to be more natural in applications, and gives a direct way to evaluate asymptotic behavior of functions relative to the given size.

We start with polynomial on average functions (Section 9.2.1), then discuss the general case (Section 9.2.2), and apply the average case analysis to the time complexity of algorithms (Section 9.2.3).

In Section 9.2.4 we compare the average case complexity with the worst-case complexity and discuss what kind of hardness the average case complexity actually measures. Finally, in Section 9.2.5 we focus on deficiencies of the average case complexity.

9.2.1. Polynomial on average functions. There are several approaches to polynomial on average functions, some of them result in equivalent definitions and some do not. Here we follow Levin's original approach [171], which has been further developed by Y. Gurevich [123] and Impagliazzo [138].

Let $f : I \rightarrow \mathbb{R}^+$ be a non-negative real function. We say that f has a polynomial upper bound with respect to the size s if there exists a real polynomial p such that $f(w) \leq p(s(w))$ for any $w \in I$, i.e.,

$$\forall w \in I_n \quad f(w) \leq p(n).$$

To get a similar condition “on average” one may integrate the inequality above, which results in the following “naive definition” of the polynomial on average functions

DEFINITION 9.2.1. A function $f : I \rightarrow \mathbb{R}^+$ is expected polynomial on spheres (with respect to an ensemble of spherical distributions $\{\mu_n\}$) if for every $n \in \mathbb{N}$,

$$\int_{I_n} f(x) \mu_n(x) \leq p(n).$$

Equivalently, f is expected polynomial on spheres if there exists $k \geq 1$ such that

$$(11) \quad \int_{I_n} f(w) \mu_n(w) = O(n^k).$$

No doubt, the functions satisfying (11) are “polynomial on μ -average” with respect to the size function s , but the class of these functions is too small. Indeed, one should expect that the class of polynomial on average functions must be closed under addition, multiplication, and multiplication by a scalar. The following example shows that the class of functions satisfying the condition (11) is not closed under multiplication.

EXAMPLE 9.2.2. Let $I = \{0, 1\}^*$ be the set of all words in the alphabet $\{0, 1\}$, $s(w) = |w|$, the length of the word, and μ is length-preserving, so μ_n is the uniform distribution on I_n . Denote by S_n a fixed subset of I_n which has $\frac{2^n - 1}{2^n}$ elements. Now we define a function $f : I \rightarrow \mathbb{N}$ by its values on each sphere I_n as follows:

$$f(w) = \begin{cases} n, & \text{if } w \in S_n, \\ 2^n, & \text{if } w \notin S_n. \end{cases}$$

Then

$$\int_I f(w) \mu(w) = n(1 - 2^{-n}) + 2^n 2^{-n} = O(n),$$

so the function f satisfies the condition (11), but its square f^2 does not:

$$\int_I f^2(w) \mu(w) = n^2(1 - 2^{-n}) + 2^{2n} 2^{-n} = O(2^n).$$

The following definition gives a wider class of polynomial on μ -average functions which is closed under addition, multiplication, and multiplication by a scalar.

DEFINITION 9.2.3 (Spherical definition). A function $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average on spheres if there exists an $\varepsilon > 0$ such that

$$(12) \quad \int_{I_n} f^\varepsilon(w) \mu_n(w) = O(n).$$

It is convenient, sometimes, to write the condition (9.2.3) in the following form (using the fact that $s(w) = n$ for $w \in I_n$):

$$\int_{I_n} \frac{f^\varepsilon(w)}{s(w)} \mu_n(w) = O(1).$$

The following proposition shows that every function which is expected polynomial on spheres satisfies the spherical definition of polynomial on μ average.

PROPOSITION 9.2.4. *Let $\{\mu_n\}$ be an ensemble of spherical distributions. If a function $f : I \rightarrow \mathbb{R}^+$ is an expected polynomial on spheres relative to $\{\mu_n\}$ then it is a polynomial on μ -average on spheres (satisfies Definition 9.2.3).*

Proof. Suppose for some $k \geq 1$

$$\int_{I_n} f(w) \mu_n(w) \leq cn^k.$$

Put $\varepsilon = \frac{1}{k}$ and denote $S_n = \{w \in I_n \mid f^\varepsilon(w) \leq s(w)\}$. Then

$$\begin{aligned} \int_{I_n} \frac{f(w)^\varepsilon}{s(w)} \mu_n(w) &= \int_{S_n} \frac{f(w)^\varepsilon}{s(w)} \mu_n(w) + \int_{\overline{S_n}} \frac{f(w)^\varepsilon}{s(w)} \mu_n(w) \\ &\leq 1 + \int_{I_n} \left(\frac{f(w)^\varepsilon}{s(w)} \right)^k \mu_n(w) \leq 1 + \int_{I_n} \frac{f(w)}{n^k} \mu_n(w) \leq 1 + c. \end{aligned}$$

Therefore,

$$\int_I \frac{f(w)^\varepsilon}{s(w)} \mu(w) = \sum_n \left(\mu(I_n) \int_{I_n} \frac{f(w)^\varepsilon}{s(w)} \mu_n(w) \right) \leq (1+c) \sum_n \mu(I_n) = 1+c.$$

Since $s(w) = n$ for $w \in I_n$ one has

$$\int_{I_n} f^\varepsilon(w) \mu_n(w) \leq (1+c)n,$$

so f is polynomial on μ -average on spheres. ■

However, this class is not wide enough yet. The example below shows a function which is “morally” polynomial on average, but does not satisfy the conditions of the spherical definition. To relax the condition (12) rewrite it first in the form

$$\int_{I_n} f^\varepsilon(w) s(w)^{-1} \mu_n(w) = O(1).$$

Recall now, that $\mu_n(w) = \mu(w)/\mu(I_n)$, hence

$$\int_{I_n} f^\varepsilon(w) s(w)^{-1} \mu(w) = O(1) \mu(I_n).$$

Therefore, for some constant $C > 0$

$$(13) \quad \int_I f^\varepsilon(w) s(w)^{-1} \mu(w) \leq \sum_n C \mu(I_n) = C.$$

The condition (13) describes a larger class of functions which are intuitively polynomial on μ -average. This condition, as well as the definition below of polynomial on average functions, is due to Levin [171].

DEFINITION 9.2.5 (Levin's definition). A function $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average if there exists $\varepsilon > 0$ such that

$$\int_I (f(w))^\varepsilon s(w)^{-1} \mu(w) < \infty.$$

It is convenient to reformulate this definition in the following equivalent form (see [16], [123])

DEFINITION 9.2.6. A function $f : I \rightarrow \mathbb{R}^+$ is linear on μ -average if

$$\int_I f(w) s(w)^{-1} \mu(w) < \infty,$$

and f is polynomial on μ -average if $f \leq p(l)$ for some linear on μ -average function $l : I \rightarrow \mathbb{R}^+$ and a polynomial p .

As we have mentioned already, the class of polynomial on μ -average functions described in Levin's definition, contains all functions which are average polynomials on spheres (the Spherical Definition 9.2.3). It is not hard to see also that this class is closed under addition, multiplication, and multiplication by a scalar. From now on, by polynomial on μ -average functions we refer to functions from Levin's definition.

In general, not every function which is polynomial on μ -average is also an average polynomial on spheres. However, if the distribution μ satisfies some sufficiently strong non-flatness conditions then the statement holds.

PROPOSITION 9.2.7 ([123]). *Let μ be an atomic distribution on I and $\{\mu_n\}$ the induced ensemble of spherical distributions. If there exists a polynomial $p(n)$ such that for every n either $\mu(I_n) = 0$ or $\mu(I_n) \geq \frac{1}{p(n)}$ then every function $f : I \rightarrow \mathbb{R}^+$ which is polynomial on μ -average is also an average polynomial on spheres (relative to the ensemble $\{\mu_n\}$).*

To formulate a useful criterion of polynomial on average functions we need one more notion. A function f is a *rarity function* if the integral

$$\int_I f(w) \mu(w)$$

(the expected value of f) converges.

PROPOSITION 9.2.8 ([123]). *A function $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average if and only if there exists a rarity function $h : I \rightarrow \mathbb{R}^+$ and a polynomial $p(x)$ such that for every $w \in I$*

$$f(w) \leq p(s(w), h(w)).$$

As a corollary we get another sufficient condition for a function f to be polynomial on μ -average, that is used sometimes as an alternative way to introduce polynomial on average functions.

COROLLARY 9.2.9. *If for some polynomial $p(x)$,*

$$\int_I \frac{f(w)}{p(s(w))} \mu(w) < \infty,$$

then the function f is polynomial on μ -average.

Proof. Indeed, in this case

$$\int_I \frac{f(w)}{s(w)p(s(w))} \mu(w) \int_I \frac{f(w)}{p(s(w))} \mu(w) < \infty,$$

so the function $l(w) = \frac{f(w)}{p(s(w))}$ is linear on μ -average. Hence $f(w) = l(w)p(s(w))$ is polynomial on μ -average. ■

One more definition of polynomial on μ -average functions was formally introduced by Impagliazzo in [137]. This is a volume analog of the spherical definition above, but contrary to the spherical case it is equivalent to Levin's definition. Impagliazzo's definition is very natural and allows one to work only with distributions on finite sets — the balls B_n of I .

DEFINITION 9.2.10 (Volume definition). Let $\{\mu_n\}$ be an ensemble of volume distributions on balls $\{B_n\}$ of I . A function $f : I \rightarrow \mathbb{R}$ is polynomial on average with respect to $\{\mu_n\}$ if there exists an $\varepsilon > 0$ such that

$$\int_{B_n} f^\varepsilon(x) \mu_n(x) = O(n).$$

It has been noticed in Section 9.1.2 that an atomic measure μ on I induces an ensemble of atomic volume distributions $\{\mu_n\}$ on balls B_n , and vice versa, every ensemble of atomic distributions $\{\mu_n\}$ on balls B_n such that μ_{n-1} is the distribution induced by μ_n on B_{n-1} , gives rise to an atomic distribution μ' on I which induces $\{\mu_n\}$. In this case one has two different definitions of a function $f : I \rightarrow \mathbb{R}^+$ to be polynomial on average relative to μ and $\{\mu_n\}$. However, the following result shows that these definitions give the same class of polynomial on average functions.

PROPOSITION 9.2.11 ([137]). *Let μ be a distribution on I and $\{\mu_n\}$ the corresponding induced ensemble of volume distributions. Then a function $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average if and only if it is polynomial on average relative to the ensemble $\{\mu_n\}$.*

Proof. Suppose $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average, so there is $\varepsilon > 0$ such that

$$\int_I f(w)^\varepsilon s(w)^{-1} \mu(w) < \infty.$$

Then

$$\begin{aligned} \int_{B_n} f(w)^\varepsilon \mu_n(w) &\leq \int_{B_n} \frac{n}{s(w)} f(w)^\varepsilon \frac{\mu(w)}{\mu(B_n)} \\ &\leq \frac{n}{\mu(B_m)} \int_{B_n} \frac{1}{s(w)} f(w)^\varepsilon \mu(w) = O(n), \end{aligned}$$

where B_m is the ball of minimal radius with $\mu(B_m) \neq 0$ (such a ball always exists). Hence, f is polynomial on average relative to $\{\mu_n\}$.

Conversely, assume now that f is polynomial on average relative to $\{\mu_n\}$, so for some $\varepsilon > 0$,

$$\int_{B_n} f(x)^\varepsilon \mu_n(x) = O(n).$$

Put $S = \{w \in I \mid f(w)^{\frac{\varepsilon}{3}} \leq s(w)\}$. Then

$$\int_I f(w)^{\frac{\varepsilon}{3}} s(w)^{-1} \mu(w) = \int_S f(w)^{\frac{\varepsilon}{3}} s(w)^{-1} \mu(w) + \int_{\overline{S}} f(w)^{\frac{\varepsilon}{3}} s(w)^{-1} \mu(w)$$

$$\begin{aligned}
&\leq \int_I \mu(w) + \int_{\overline{S}} \frac{f(w)^\varepsilon}{f(w)^{\frac{2\varepsilon}{3}} s(w)} \mu(w) \leq 1 + \sum_n \int_{I_n} \frac{f(w)^\varepsilon}{n^3} \mu(w) \\
&\leq 1 + \sum_n \frac{1}{n^3} \int_{B_n} f(w)^\varepsilon \mu(w) \leq 1 + \sum_n \frac{1}{n^3} \int_{B_n} f(w)^\varepsilon \mu_n(w) \\
&= 1 + \sum_n \frac{O(n)}{n^3} < \infty
\end{aligned}$$

and the leftmost integral above converges, as required. \blacksquare

The results of this section show that the class of polynomial on average functions is very robust and, it seems, it contains all the functions that naturally look like “polynomial on average”.

9.2.2. Average case behavior of functions. Definition 9.2.6 allows one to introduce general complexity classes on average.

DEFINITION 9.2.12. Let $f : I \rightarrow \mathbb{R}$ and $t : \mathbb{R} \rightarrow \mathbb{R}$ be two functions. Then f is t on μ -average if $f(w) = t(l(x))$ for some linear on μ -average function l .

This definition can be reformulated in a form similar to Levin’s Definition 9.2.5. To this end suppose for simplicity that a function $t : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is continuous, unbounded and monotone increasing, or $t : \mathbb{N}^+ \rightarrow \mathbb{N}^+$ and it is injective and unbounded. In the latter case, define for $x \in \mathbb{N}$,

$$t^{-1}(x) = \min\{y \mid t(y) \geq x\}.$$

Then it is not hard to show that a function $f : I \rightarrow \mathbb{R}$ is t on μ -average if and only if

$$\int_I t^{-1}(f(x)) s(x)^{-1} \mu(x) < \infty.$$

For a function t one can consider the class $Ave_\mu(t)$ of functions $f : I \rightarrow \mathbb{R}$ which are t on μ -average. By varying the function t one may expect to get a hierarchy of functions on μ -average. However, it is not always the case.

EXAMPLE 9.2.13. If a function $f : I \rightarrow \mathbb{R}$ is a^x on μ -average for some $a > 1$, then f is b^x on μ -average for any $b > 1$.

Proof. Indeed,

$$\int_I \frac{\log_b(f(x))}{s(x)} \mu(x) = \frac{1}{\log_b a} \int_I \frac{\log_a(f(x))}{s(x)} \mu(x) < \infty,$$

so f is b^x on μ -average. \blacksquare

We will have more to say on the hierarchy of time complexity of algorithmic problems in Section 9.2.3.

9.2.3. Average case complexity of algorithms. In this section we apply the average case analysis of functions to average case behavior of algorithms and time complexity of algorithmic problems.

Let \mathcal{D} be a stratified distributional algorithmic problem, $I = I_{\mathcal{D}}$ — the set of instances of \mathcal{D} equipped with a size function $s = s_{\mathcal{D}} : I \rightarrow \mathbb{N}$, and an atomic probability distribution $\mu = \mu_{\mathcal{D}}$.

If \mathcal{A} is a total decision algorithm for \mathcal{D} then one can estimate overall efficiency of the algorithm \mathcal{A} by its *expected running time*:

$$\int_I T_{\mathcal{A}}(w) \mu(w).$$

Another way to characterize the “average” behavior of \mathcal{A} relative to the size of inputs comes from the average case analysis of the time function $T_{\mathcal{A}}$.

An algorithm \mathcal{A} has *polynomial time on μ -average* if $T_{\mathcal{A}} : I \rightarrow \mathbb{N}$ is polynomial on μ -average function. We say that \mathcal{A} has *polynomial time upper bound on μ -average* if $T_{\mathcal{A}}(x) \leq f(x)$ for some polynomial on μ -average function $f : I \rightarrow \mathbb{N}$. Similarly, \mathcal{A} has *time upper bound $t(x)$ on μ -average* if $T_{\mathcal{A}}$ has an upper bound which is t on μ -average. These notions allow one to introduce complexity classes of algorithmic problems.

DEFINITION 9.2.14. Let \mathcal{D} be a stratified distributional problem. Then:

- 1) \mathcal{D} is *decidable in polynomial time on average (AvePTime)* if there exists a polynomial time on μ -average decision algorithm \mathcal{A} for \mathcal{D} . The class of stratified distributional problems decidable in AvePTime is denoted by **AvePTime** (or simply by **AveP**).
- 2) \mathcal{D} is *decidable in t time on average (AveTime(t))* for some time bound t if there exists a decision algorithm for \mathcal{D} with a time upper bound t on μ -average. The class of stratified distributional problems decidable in AveTime(t) is denoted by **AveTime(t)**.

9.2.4. Average case vs. worst case. In this section we compare the average-case complexity to the worst-case one. To do this we briefly discuss what kind of “hardness” the average case complexity does measure, indeed. We refer to Gurevich’s paper [125] for a detailed discussion of these issues.

The main outcome of the development of the average-case complexity is that this type of complexity gives a robust mathematical framework and provides one with a much more balanced view-point on the computational hardness of algorithmic problems than the worst-case complexity. It turns out, for example, that many algorithmic problems that are hard in the worst-case are easy on average.

Consider, for instance, the Hamiltonian Circuit Problem (HCP). Recall that a *Hamiltonian circuit* in a graph is a closed path that contains every vertex exactly once. HCP asks for a given finite graph Γ to find a Hamiltonian circuit in Γ . It is known that this problem is **NP**-complete (see, for example, [89]). However, Gurevich and Shelah showed in [126] that there is a decision algorithm for the HCP which is linear time on average. More precisely, if one has a distribution on finite graphs such that any two vertices u, v in a given random finite graph Γ are connected by an edge with a fixed uniform probability $\alpha, 0 < \alpha < 1$, then there is an algorithm that solves the problem in time linear on average.

COROLLARY 9.2.15. *There are **NP**-complete problems which are polynomial on average with respect to some natural distributions.*

9.2.5. Average case behavior as a trade-off. The concept of average case behavior of functions may look sometimes counterintuitive. In this section we discuss what kind of knowledge the average case analysis of functions or algorithms brings to the subject. We argue that the average case analysis is a wrong tool when one tries to describe the “typical behavior” of a function, say its behavior on “most” or “typical” inputs. We follow Gurevich [123, 125] and Impagliazzo [137] in the discussion on the essence of the concept of a polynomial on average function. At the end of the section we give a convenient tool to get general upper bounds on average.

To indicate the typical problems with the average case behavior we give two examples.

EXAMPLE 9.2.16. Let $I = \{0, 1\}^*$ be the set of all words in the alphabet $\{0, 1\}$. Define the size of a word $w \in I$ to be equal to the length $|w|$ of the word, and let the distribution μ on I be such that its restriction on a sphere I_n is uniform for every n . Denote by S_n a subset of I_n of measure $\frac{2^n - 1}{2^n}$, so S_n has precisely $2^n - 1$ elements. Now we define a function $l : I \rightarrow \mathbb{N}$ by its values on each sphere I_n as follows:

$$l(w) = \begin{cases} 0, & \text{if } w \in S_n, \\ 2^n, & \text{if } w \notin S_n. \end{cases}$$

Then

$$\int_I l(w)|w|^{-1}\mu(w) < \infty,$$

so the function l is linear on μ -average. Now we define a function $f : I \rightarrow \mathbb{N}$ by its values on each sphere I_n by:

$$f(w) = \begin{cases} 1, & \text{if } w \in S_n, \\ 2^{2^n}, & \text{if } w \notin S_n. \end{cases}$$

Clearly, $f(w) = 2^{l(w)}$, so f is exponential on μ -average. However, $f(w) = 1$ on most words $w \in I$.

Another type of a strange behavior of functions polynomial on average is described in the following example.

EXAMPLE 9.2.17. Let $I = \{0, 1\}^*$. Define the size of a word $w \in I$ to be equal to the length $|w|$ of the word, and let the distribution μ on I be given by

$$\mu(w) = 2^{-2|w|-1}.$$

If $f : I \rightarrow \mathbb{N}$ is a function such that $f(w) = 2^{|w|}$. Then

$$\sum_{|w| \neq 0} \frac{f^\varepsilon(w)}{|w|} \mu(w) = \sum_{n=1}^{\infty} \sum_{|w|=n} \frac{2^{n\varepsilon}}{2^{2n+1}} = \sum_{n=1}^{\infty} 2^{n\varepsilon - n - 1},$$

which converges for any $\varepsilon < 1$. Hence f is polynomial on μ -average. On the other hand, the function $f(x)$ is clearly “exponential on most inputs”.

To properly understand the examples above one has to focus on a general question:

What kind of knowledge does the average case analysis of behavior of functions bring into the subject?

In [125] Gurevich explains, in terms of a Challenger-Solver game, what kind of hardness the concept of the average case complexity is aiming to. First of all, the examples above show that it is not the cost for the Solver to win the game on the most typical inputs, or even expected cost, rather, it captures the *trade-off* between a measure of difficulty and the fraction of hard instances of the problem. Thus, to have polynomial on average time an algorithm should have only a sub-polynomial fraction of inputs that require superpolynomial time to compute.

The following definition (due to Impagliazzo [137]) is a rigorous attempt to grasp this type of trade-offs to stay in the class of polynomial on μ -average functions.

Let \mathcal{D} be a stratified distributional algorithmic problem, $I = I_{\mathcal{D}}$, $s = s_{\mathcal{D}}$, and $\mu = \{\mu_n\}$ is an ensemble of volume distributions for I . A partial algorithm \mathcal{A} solves \mathcal{D} with *benign faults* if for every input $w \in I$ it either outputs the correct answer or it stops and says “?” (“Do not know”). A polynomial time *benign algorithm scheme* for \mathcal{D} is an algorithm $\mathcal{A}(w, \delta)$ such that for any input $w \in I$ and $\delta \in \mathbb{N}$ the algorithm \mathcal{A} does the following:

- there is a polynomial $p(x, y)$ such that \mathcal{A} stops on (w, δ) in time $p(s(w), \delta)$;
- \mathcal{A} solves \mathcal{D} on input w with benign faults;
- for any $\delta \in \mathbb{N}$ and $n \in \mathbb{N}$ $\mu_n(\{w \in B_n \mid A(w, \delta) = ?\}) \leq \frac{1}{\delta}$.

The following results shows equivalence of polynomial on average problems and existence of polynomial time benign algorithm schemes.

LEMMA 9.2.18 (Impagliazzo [137]). *A stratified distributional algorithmic problem \mathcal{D} with an ensemble of volume distributions $\mu = \{\mu_n\}$ is polynomial on μ -average if and only if \mathcal{D} has a polynomial time benign decision algorithm scheme $\mathcal{A}(w, \delta)$.*

Now we can properly analyze the examples above.

The Example 9.2.16 reveals “traces” of the worst-case behavior in the average case complexity. Indeed, in this example the balance between the time complexity of an algorithm on the worst-case inputs and the measure of the fraction of these inputs is broken, i.e., the fraction of hard instances is small but the time required by the algorithm to finish computation on these instances is very large (compare to the size of the fraction), therefore the average case complexity does not reflect behavior of the algorithm on the most inputs.

In Example 9.2.17, the balance between the fraction of the hard inputs and the measure of this fraction is almost perfect, therefore the function “looks polynomial” on average, which is, of course, not the case.

The cause of this discrepancy is reflected in the notion of a flat distribution (see e.g. [123]). One may try to avoid such distributions altogether, but they appear quite naturally in many typical situations (see [123, 125] and [29]), so it is difficult to ignore them. On the other hand, if one puts into the game a natural non-flatness condition, say as in Proposition 9.2.7, then polynomial on average functions become polynomial on average on spheres, that makes the class of polynomial on average functions smaller. In any case, Example 9.2.17 is troubling.

To deal with the trade-off in the general case we introduce the following definitions. A function $h : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$ is μ -balanced on a subset $K \subseteq I$ if

$$\sum_n h(n)n^{-1}\mu(K \cap I_n) < \infty.$$

A function $U : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is an s -upper bound for $f : I \rightarrow \mathbb{R}^+$ on a subset $K \subseteq I$ if $f(w) \leq U(s(w))$ for every $w \in K$. Finally, f is (s, μ) -balanced on K if there exists an s -upper bound U for f on K which is μ -balanced on K , i.e.,

$$\sum_n U(n)n^{-1}\mu(K \cap I_n) < \infty.$$

PROPOSITION 9.2.19 (Balance test). *Let $f : I \rightarrow \mathbb{R}^+$ and $f_1 : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be two functions. Suppose f_1 is a bijection. If there exists a subset $K \subseteq I$ such that*

- 1) f is f_1 on μ -average on K (in the μ -induced measure μ_K on K),
- 2) $f_1^{-1}(f)$ is (s, μ) -balanced on $I - K$,

then f is f_1 on μ -average.

Proof. To show that f is f_1 on μ average it suffices prove that the integral

$$\int_I \frac{f_1^{-1}(f(w))}{s(w)} \mu(w)$$

converges. To see this consider

$$\begin{aligned} \int_I \frac{f_1^{-1}(f(w))}{s(w)} \mu(w) &= \int_K \frac{f_1^{-1}(f(w))}{s(w)} \mu(w) + \int_{I-K} \frac{f_1^{-1}(f(w))}{s(w)} \mu(w) \\ &\leq \mu(K) \int_K \frac{f_1^{-1}(f(w))}{s(w)} \mu_K(w) + \int_{I-K} \frac{U(s(w))}{s(w)} \mu(w) \\ &\leq \mu(K) \int_K \frac{f_1^{-1}(f(w))}{s(w)} \mu_K(w) + \Sigma_n \frac{U(n)}{n} \mu((I-K) \cap I_n) < \infty. \end{aligned}$$

(here U is an s -upper bound on $I-K$ for $f_1^{-1}(f)$ which is (s, μ) -balanced on $I-K$)

This shows that $T_{\mathcal{A}}$ runs within the bound f_1 on μ -average, as claimed. \blacksquare

REMARK 9.2.20. The condition on f_1 to be a bijection is introduced, just to ensure that $f_1^{-1}(f(w))$ always exists. One can easily replace it, if needed, with a more relaxed one. In the case of algorithms their time functions are integer functions of the type $f : \mathbb{N} \rightarrow \mathbb{N}$, in which case it suffices (to get a result similar to the Balance test) to assume that $f_1 : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ is unbounded and define $f_1^{-1}(n) = \min\{m \mid f_1(m) \geq n\}$.

An adaptation of the argument above to the time functions of algorithms gives a robust test for an algorithm to run within a given upper bound on μ -average. Below we assume that $f_1 : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ is unbounded and f^{-1} is defined as above.

PROPOSITION 9.2.21 (Average running time test). *Let \mathcal{A} be an algorithm with a set of inputs I . If there exists a subset $K \subseteq I$ such that*

- 1) \mathcal{A} runs on K within an upper bound f_1 on μ -average (in the μ -induced measure on K),
- 2) there exists an upper bound f for \mathcal{A} on $I - K$ such that $f_1^{-1}(f)$ is (s, μ) -balanced on $I - K$,

then \mathcal{A} runs on I within the time bound f_1 on μ -average.

Proof. Observe, that $T_{\mathcal{A}}(w) \leq f_1(s(w))$ if $w \in K$ and $T_{\mathcal{A}}(w) \leq f(s(w))$ otherwise. We claim that the algorithm \mathcal{A} runs in time f_1 on μ -average. To see this consider

$$\begin{aligned} \int_I \frac{f_1^{-1}(T_{\mathcal{A}}(w))}{s(w)} \mu(w) &= \int_K \frac{f_1^{-1}(T_{\mathcal{A}}(w))}{s(w)} \mu(w) + \int_{I-K} \frac{f_1^{-1}(T_{\mathcal{A}}(w))}{s(w)} \mu(w) \\ &\leq \int_K \frac{f_1^{-1}(f_1(s(w)))}{s(w)} \mu(w) + \int_{I-K} \frac{f_1^{-1}(f(w))}{s(w)} \mu(w) \\ &\leq \mu(K) + \int_{I-K} \frac{U(s(w))}{s(w)} \mu(w) < \infty. \end{aligned}$$

This shows that $T_{\mathcal{A}}$ runs within the bound f_1 on μ -average, as claimed. \blacksquare

REMARK 9.2.22. It is easy to show particular examples of K , f , and f_1 to satisfy the conditions of the Proposition 9.2.21. Some of them are given in Corollary 10.2.5 of Section 10.2.3.

9.2.6. Deficiency of the average case complexity. In this section we argue that despite the fact that the average case complexity embodies a much more practical viewpoint on complexity of algorithms (than the worst-case complexity) it does not, in general, provide the type of analysis of algorithms which is required in many applications where the focus is on behavior of algorithms on most or typical inputs. For example, in cryptography the average case analysis does not play much of a role, since (see Example 9.2.16) an algorithm, that lies at the heart of the algorithmic security of a cryptosystem, can be of exponential time on average (so presumably hard), and have linear time on most inputs. Another typical example is the Dantzig's algorithm, mentioned above, for the linear programming, which is much more robust in applications than the existing polynomial time (in the worst case!) algorithms. The main reason for the inadequate results of the average case analysis in many situations is, of course, that there are natural algorithms which have high (say, exponential) average case complexity, but are still very fast on most inputs; conversely, there is obviously exponential time on most input functions which are polynomial on average relative to some quite natural distributions. In fact, it seems, it is not easy to come up with any real application which would require the average case analysis of algorithms, rather than the worst-case analysis, or the analysis on the “most inputs”.

One of the ways to introduce a complexity class that would reflect the behavior of algorithms on most inputs comes from statistics. The idea is to use the median time of the Solver, since the median better represents the typical behavior. However, it is not clear at all how robust such a theory would be, and there are indications that difficulties similar to the average case naturally occur.

Our main point in this discussion is that quite often it would be convenient and more practical to work with another type of complexity which focuses on generic inputs of algorithms, ignoring sparse sets of “non-typical” inputs. This type of complexity, coined as *generic case complexity*, has emerged in asymptotic mathematics for some time now, and it is quite close, in spirit, to the above mentioned results of Vershik and Sporyshev [280], and Smale [260] in their analysis of the simplex algorithm for linear programming problems.

CHAPTER 10

Generic Case Complexity

10.1. Generic Complexity

In this section we discuss a general notion of generic complexity of algorithms and algorithmic problems.

10.1.1. Generic sets. We start by introducing some necessary notation.

Let I be a set. Denote by $\mathcal{P}(I)$ the subset algebra of I , i.e., the set of all subsets of M with operations of union, intersection and complementation.

A *pseudo-measure* on I is a real-valued non-negative function $\mu : \mathcal{A} \rightarrow \mathbf{R}^+$ defined on a subset $\mathcal{A} \subset \mathcal{P}(I)$ such that

- 1) \mathcal{A} contains M and is closed under disjoint union and complementation;
- 2) $\mu(I) = 1$ and for any disjoint subsets $A, B \in \mathcal{A}$,

$$\mu(A \cup B) = \mu(A) + \mu(B).$$

In particular, $\mu(\overline{A}) = 1 - \mu(A)$.

If \mathcal{A} is a subalgebra of $\mathcal{P}(I)$ then μ is a measure. We make a point to consider pseudo-measures here since the asymptotic density function (see Section 10.1.2), which is one of the fundamental tools in asymptotic mathematics, in general, is only a pseudo-measure, not a measure.

A pseudo-measure μ is called *atomic* if $\mu(Q)$ is defined for any finite subset Q of I .

DEFINITION 10.1.1. Let μ be a pseudo-measure on I . We say that a subset $Q \subseteq I$ is *generic* if $\mu(Q) = 1$ and *negligible* if $\mu(Q) = 0$.

The following lemma is easy.

LEMMA 10.1.2. *Let I be a set equipped with a pseudo-measure μ . The following holds for arbitrary subsets S, T of I :*

- 1) *S is generic if and only if its complement \overline{S} is negligible.*
- 2) *If S is generic and $S \subseteq T$ then T is generic.*
- 3) *Finite unions and intersections of generic (negligible) sets is generic (negligible).*
- 4) *If S is generic and T is negligible, then $S - T$ is generic.*
- 5) *The set \mathcal{B} of all generic and negligible sets forms an algebra of subsets of I .*
- 6) *$\mu : \mathcal{B} \rightarrow \mathbf{R}^+$ is a measure on I .*

If the set I is countable and μ is an atomic probabilistic measure on I with no elements of probability zero (for example, Exponential or Cauchy distributions from Section 9.1.1), then I is the only generic set in I . In this situation the notions of generic and negligible sets are not very interesting. However, if the set I has a

natural stratification, then the measure μ gives rise to a very useful pseudo-measure on I — the asymptotic density ρ_μ relative to μ . We discuss this in the next section.

10.1.2. Asymptotic density. Let I be a set with a size function $s : I \rightarrow \mathbb{N}^+$ (or $s : I \rightarrow \mathbb{R}^+$). Let $\mu = \{\mu_n\}$ be an ensemble of spherical distributions for I (so μ_n is a distribution on the sphere I_n).

For a set $R \subseteq I$ one can introduce the *spherical asymptotic density* ρ_μ , with respect to the ensemble μ , as the following limit (if it exists):

$$\rho_\mu(R) = \lim_{n \rightarrow \infty} \mu_n(R \cap I_n).$$

The most typical example of an asymptotic density arises when all the spheres I_n are finite.

EXAMPLE 10.1.3. Suppose the spheres I_n are finite for every $n \in \mathbb{N}$. Denote by μ_n the uniform distribution on I_n . Then for a subset $R \subseteq I$,

$$\mu_n(R) = \frac{|R \cap I_n|}{|I_n|},$$

is just the frequency of occurrence of elements from R in the sphere I_n . In this case the spherical asymptotic density $\rho_\mu(R)$ is just the standard *uniform spherical asymptotic density* of R (see Section 3.2.3). Usually, we denote $\rho_\mu(R)$ by $\rho(R)$, and $\mu_n(R)$ by $\rho_n(R)$.

The following example shows that for some sets R the asymptotic density $\rho(R)$ does not exist.

EXAMPLE 10.1.4. Let $I = X^*$ be the set of all words in a finite alphabet $X = \{x_1, \dots, x_m\}$ and R the subset of all words of even length. Then $\rho_n(R) = 1$ for even n and $\rho_n(R) = 0$ for odd n , so the limit $\rho(R)$ does not exist.

One way to guarantee that the asymptotic density $\rho(R)$ always exists is to replace $\lim_{n \rightarrow \infty} \rho_n(R)$ with the $\limsup_{n \rightarrow \infty} \rho_n(R)$. In the sequel, we mostly use the standard limit (since the results are stronger in this case), and only occasionally — the upper limit (when the standard limit does not exist).

Another way to make the asymptotic density smoother is to replace spheres I_n in the definition above with the balls $B_n = B_n(I)$. If $\nu = \{\nu_n\}$ is an ensemble of volume distributions for I then the *volume* (or *disc*) asymptotic density ρ_ν^* relative to ν is defined for a subset $R \subseteq I$ as the limit (if it exists):

$$\rho_\nu^*(R) = \lim_{n \rightarrow \infty} \nu_n(R).$$

The following lemma is obvious.

LEMMA 10.1.5. *The spherical and volume asymptotic densities on I are pseudo-measures on I .*

Now we consider another typical way to define asymptotic densities of sets. As above, let I be a set with a size function $s : I \rightarrow \mathbb{N}$. Suppose μ is a probability distribution on I such that all the spheres I_n are measurable subsets of I . Denote by μ_n the probability distribution on I_n induced by μ . In this case if R is a measurable subset of I then the frequencies $\mu_n(R)$ are well defined, so it makes sense to consider the spherical asymptotic density $\rho_\mu(R)$ or R . Observe, that in this case the balls B_n are also measurable, as finite disjoint unions of n spheres $B_n(I) = I_1 \cup \dots \cup I_n$, so the volume frequencies are also defined for R , and the notion of the volume asymptotic

density also makes sense. In particular, the probability distributions defined in Section 9.1.1 give rise to interesting asymptotic densities with non-trivial generic and negligible sets, as was mentioned before.

REMARK 10.1.6. If the probability distribution μ on I is size-invariant, i.e., for any $u, v \in I$ $\mu(u) = \mu(v)$ provided $s(u) = s(v)$, then it induces the uniform distribution on I_n , so the spherical asymptotic density with respect to μ on I is equal to the standard uniform spherical asymptotic density on I . The same is true for volume size-invariant distributions.

At the end of this section we prove a lemma which shows that the standard uniform spherical and volume asymptotic densities are equal in some natural situations.

LEMMA 10.1.7. *Let all the spheres I_n be finite and non-empty. If the standard spherical density (as in Example 10.1.3) $\rho(R)$ exists for a subset R of I then the standard volume density $\rho^*(R)$ also exists and $\rho^*(R) = \rho(R)$.*

Proof. Set $x_n = |R \cap B_n|$ and $y_n = |B_n|$. Then $y_n < y_{n+1}$ and $\lim y_n = \infty$. By Stolz's theorem

$$\rho^*(R) = \lim_{n \rightarrow \infty} \frac{x_n}{y_n} = \lim_{n \rightarrow \infty} \frac{x_n - x_{n-1}}{y_n - y_{n-1}} = \lim_{n \rightarrow \infty} \frac{|R \cap I_n|}{|I_n|} = \rho(R),$$

as claimed. ■

10.1.3. Convergence rates. Let I be a set with a size function $s : I \rightarrow \mathbb{N}$. Suppose for each $n \in \mathbb{N}$ the sphere I_n and the ball B_n are equipped with probability distributions μ_n and μ_n^* , correspondingly. The asymptotic densities ρ_μ and ρ_μ^* are defined and, by Lemma 10.1.5, they are pseudo-measures. Hence the results from Section 10.1.1 apply and the notion of a generic and a negligible set relative to ρ_μ and ρ_μ^* are defined. We would like to point out here that these asymptotic densities not only allow one to distinguish between “large” (generic) and “small” (negligible) sets, but they provide much finer methods to describe asymptotic behavior of sets at “infinity” with respect to the given size function s . In this section we introduce the required machinery.

We start with the spherical asymptotic density ρ_μ . One can introduce similar notions for the volume density ρ_μ^* ; we leave it to the reader.

Let $\mu = \{\mu_n \mid n \in \mathbb{N}\}$ be a fixed ensemble of spherical distributions for I .

DEFINITION 10.1.8. Let R be a subset of I for which the asymptotic density $\rho_\mu(R)$ exists. A function $\delta_R : n \rightarrow \mu_n(R \cap I_n)$ is called a *frequency function* of R in I with respect to the spherical ensemble μ . Its dual $\delta_{\bar{R}} : n \rightarrow \mu_n(\bar{R} \cap I_n)$ is called the *residual probability function* for R .

The density function δ_R may show how quickly the frequencies $\mu_n(R \cap I_n)$ converge to the asymptotic density $\rho(R)$ (if it exists), henceforth the convergence rate of δ_R gives a method to differentiate between various generic or negligible sets.

DEFINITION 10.1.9. Let $R \subseteq I$ and δ_R be the frequency function of R . We say that R has asymptotic density $\rho(R)$ with the convergence rate

- 1) of degree n^k if $|\rho(R) - \delta_R(n)| \sim cn^{-k}$ for some constant c ;
- 2) faster than n^{-k} if $|\rho(R) - \delta_R(n)| = o(n^{-k})$;
- 3) superpolynomial if $|\rho(R) - \delta_R(n)| = o(n^{-k})$ for any natural k ;

- 4) *subexponential* if $|\rho(R) - \delta_R(n)| = o(r^n)$ for every $r > 1$.
- 5) *exponential* if $|\rho(R) - \delta_R(n)| \sim c^n$ for some $0 < c < 1$.

Of course, one can introduce different degrees of exponential convergence, superexponential convergence, etc.

Now one can distinguish various generic (negligible) sets with respect to their convergence rates. Thus, we say that R is *superpolynomially (exponentially) generic* if it is generic and its convergence rate is superpolynomial (exponential). Sometimes we refer to superpolynomial generic sets as *strongly generic* sets. Note that in the original papers [146, 147] the strongly generic sets meant exponentially generic sets, but it turned out that in most applications it is more convenient to have a special name for superpolynomial generic sets. We define *superpolynomial* and *exponentially negligible* sets as the complements of the corresponding generic sets.

EXAMPLE 10.1.10. Let $I = X^*$ be the set of all words in a finite alphabet X with the length of the word as the size function and with the standard uniform distribution on spheres I_n . If $R = wX^*$ is the cone generated by a given word $w \in I$ then $\rho(R) = |X|^{-|w|}$, so R is neither generic nor negligible.

10.1.4. Generic complexity of algorithms and algorithmic problems. In this section we introduce generic complexity of algorithms and algorithmic problems. For simplicity we consider only spherical distributions here, leaving it to the reader to make obvious modifications in the case of volume distributions.

We start with a general notion of generic decidability of distributional algorithmic (decision or search) problems.

DEFINITION 10.1.11. Let \mathcal{D} be a distributional computational problem. A partial decision algorithm \mathcal{A} for \mathcal{D} *generically solves* the problem \mathcal{D} if the halting set $H_{\mathcal{A}}$ of \mathcal{A} is generic in $I = I_{\mathcal{D}}$ with respect to the given probability distribution $\mu = \mu_{\mathcal{D}}$ on I . In this case we say that \mathcal{D} is *generically decidable*.

Note that a priori, we do not require that a generically decidable problem be decidable in the worst case.

To discuss generic complexity of algorithms and algorithmic problems one needs, as usual, to have a size function $s : I \rightarrow \mathbb{N}$ on the set of inputs $I = I_{\mathcal{D}}$. Let $\mu = \{\mu_n\}$ be an ensemble of spherical distributions on I relative to the size function s . Let \mathcal{A} be a partial decision algorithm for \mathcal{D} with the halting set $H_{\mathcal{A}}$, and a partial time function $T_{\mathcal{A}} : I \rightarrow \mathbb{N}$ (see Section 3.3 for definitions). We assume here that if \mathcal{A} does not halt on $x \in I$, then $T_{\mathcal{A}}(x) = \infty$.

DEFINITION 10.1.12. Let \mathcal{D} be a stratified distributional computational problem and \mathcal{A} a partial decision algorithm for \mathcal{D} . A time function $f(n)$ is a *generic upper bound* for \mathcal{A} if the set

$$H_{\mathcal{A},f} = \{w \in I \mid T_{\mathcal{A}}(w) \leq f(s(w))\}$$

is generic in I with respect to the spherical asymptotic density ρ_{μ} . In this case, the residual probability function

$$\mathcal{C}_{\mathcal{A},f}(n) = 1 - \rho_n(H_{\mathcal{A},f}) = 1 - \mu_n(H_{\mathcal{A},f} \cap I_n)$$

is termed the *control sequence* of the algorithm \mathcal{A} relative to f .

Similarly, $f(n)$ is a *strongly generic* (*exponentially generic*, etc.) time upper bound for \mathcal{A} if the set $H_{\mathcal{A},f}$ is strongly generic (has exponential convergence rate, etc.).

Now we are ready to define generic complexity classes of algorithmic problems.

DEFINITION 10.1.13. We say that a stratified distributional decision problem \mathcal{D} is:

- *decidable generically in polynomial time* (or *GPtime decidable*) if there exists a decision algorithm \mathcal{A} for \mathcal{D} with a generic polynomial upper bound $p(n)$;
- *decidable strongly generically in polynomial time* (or *SGPtime decidable*) if there exists a decision algorithm \mathcal{A} for \mathcal{D} with a strongly generic polynomial upper bound $p(n)$.

In the situation above we say that \mathcal{D} has *generic* (*strongly generic*) *time complexity* at most $p(n)$.

By **GenP** and **Gen_{str}P** we denote the classes of problems decidable generically and, respectively, strongly generically, in polynomial time.

Similarly, one can introduce generic complexity classes for arbitrary time bounds. Furthermore, it is sometimes convenient to specify the density functions δ for the generic sets involved.

DEFINITION 10.1.14 (Generic Deterministic Time). Let $f : \mathbb{N} \rightarrow \mathbb{R}$. By **Gen_δTIME**(f) we denote the class of stratified distributional problems (\mathcal{D}, μ) such that there exists a partial decision algorithm \mathcal{A} for \mathcal{D} whose set $H_{\mathcal{A},f}$ is generic (relative to the spherical ρ , or volume ρ^* , asymptotic density with respect to μ) in I with the density function δ .

10.1.5. Deficiency of the generic complexity. In this section we discuss deficiencies of the generic complexity. One obvious deficiency is common to all complexity classes of distributional algorithmic problems. It comes from a choice of measure: if the measure is unnatural, the results could be counterintuitive. Consider the following example.

EXAMPLE 10.1.15. Let $I = \{0,1\}^*$. Define the size of a word $w \in I$ to be equal to the length $|w|$ of the word. Suppose for each n one has a disjoint decomposition of the sphere I_n into two subsets $I_n = I'_n \cup I''_n$ of size 2^{n-1} . Put

$$I' = \bigcup_n I'_n, I'' = \bigcup_n I''_n$$

and define a function $f : I \rightarrow \mathbb{N}$ by

$$f(w) = \begin{cases} 0 & \text{if } w \in I', \\ 2^{|w|} & \text{if } w \in I''. \end{cases}$$

Then one can construct two measures μ' and μ'' (which are positive on non-empty words) such that f is generically exponential with respect to the asymptotic density relative to μ' and generically constant with respect to the asymptotic density relative to μ'' . Indeed, define μ' , such that $\mu'(I'_n) = \frac{2^{n+1}-1}{2^{n+1}}$ and $\mu'(I''_n) = \frac{1}{2^{n+1}}$. Then I' is generic with respect to the asymptotic density $\rho_{\mu'}$ (so I'' is negligible). Similarly, one can define μ'' such that I'' becomes generic (and I' becomes negligible).

The only recipe here to avoid bad examples is to choose the initial measure that reflects the nature of the problem.

10.2. Generic versus average case complexity

In Section 10.2.1 below the average case complexity is compared to generic case complexity. We claim that generic case complexity is a better tool for describing behavior of algorithms on most inputs, so it is more useful in some typical applications.

It turns out however that despite all the differences, the generic and average case complexities in many particular cases are quite close to each other.

In Section 10.2.2 we show that in many cases if an algorithm is easy on average, then it is also easy generically. The opposite is not true, in fact there are exponential on average algorithms that are polynomial generically.

In Section 10.2.3 we give a robust test that shows when a generic upper time bound of an algorithm is also a time bound on average.

10.2.1. Comparing generic and average case complexities. In this section we discuss principal differences between the average and generic case complexities.

One obvious distinction is that in the generic case complexity one can consider undecidable problems as well as decidable ones. Moreover, the approach is uniform to both types of problems. For example, in [127] the generic case complexity of the Halting Problem for Turing machines is studied.

Another difference is that the generic case complexity aims at the typical behavior of the algorithm, i.e., the behavior on most inputs, rather than on the expected time, or the trade-off between the fraction of the hard inputs and the computation time of the algorithm on these inputs.

One really big advantage of the generic case complexity is that for a given problem it is much easier to find (and prove) a fast generic algorithm, then to find (and prove!) a fast algorithm on average. We refer to Chapter 11 for support of this claim.

We would also like to note that generic complexity is much clearer conceptually than the average case complexity. Indeed, if the measure is fixed, then generic case complexity describes precisely what it claims to describe, i.e., the behavior of a function on “most” inputs (i.e., on a generic set of inputs), while the average case complexity measures an allusive “trade-off”, which is not easy to formalize, let alone to measure.

Finally, we claim that what counts in most applications is precisely the generic behavior of algorithms; see the discussion in Section 9.2.6.

10.2.2. When average polynomial time implies generic. We have seen examples (see Example 9.2.17 of Chapter 9) where polynomial on average functions are not generically polynomial. However, if we restrict the class of polynomial on average functions to the narrower class of functions which are polynomial on average on spheres (see Definition 9.2.1 in Section 9.2.1), then all functions in this class are generically polynomial. This shows that, perhaps, Levin’s class of polynomial on average functions [171] is too wide.

Let \mathcal{D} be a stratified distributional algorithmic problem with a set of instances I equipped with a size function $s : I \rightarrow \mathbb{N}$ and an atomic distribution μ . It has been

noted in Section 9.1.2 that an atomic measure on I induces an ensemble of atomic distributions $\{\mu_n\}$ on spheres I_n . Recall that a function $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average on spheres if there exists $k \geq 1$ such that

$$\int_{I_n} f^{\frac{1}{k}}(w) \mu_n(w) = O(n).$$

PROPOSITION 10.2.1. *If a function $f : I \rightarrow \mathbb{R}^+$ is polynomial on μ -average on spheres, then f is generically polynomial relative to the asymptotic density ρ_μ .*

Proof. If f is an expected polynomial, then there exist a constant c and $k \geq 1$ such that for any n ,

$$\int_{I_n} f^{\frac{1}{k}}(w) \mu_n(w) \leq cn.$$

It follows that for any polynomial $q(n)$,

$$\mu_n\{x \in I_n \mid f^{\frac{1}{k}}(x) > q(n)cn\} \leq 1/q(n),$$

Now let $S(f, q, k) = \{x \in I \mid f(x) \geq (cq(s(x))s(x))^k\}$ be the set of those instances from I on which $f(x)$ is not bounded by $(cq(s(x))s(x))^k$. Then

$$\mu_n(I_n \cap S(f, q, k)) = \mu_n\{x \in I_n \mid f^{\frac{1}{k}}(x) > q(n)cn\} \leq 1/q(n).$$

Therefore, the asymptotic density ρ_μ of $S(f, q, k)$ exists and is equal to 0. This shows that f is generically bounded by the polynomial $(cq(n)n)^k$. ■

Proposition 10.2.1 gives a large class of polynomial on average functions that are generically polynomial.

COROLLARY 10.2.2. *Let \mathcal{A} be a decision algorithm for the algorithmic problem \mathcal{D} . If the expected time of \mathcal{A} (with respect to the spherical distributions) is bounded by a polynomial, then \mathcal{A} GPtime decides \mathcal{D} . Moreover, increasing the generic time-complexity of \mathcal{D} we can achieve any control sequence converging to 0 polynomially fast.*

The result of Proposition 10.2.1 can be generalized to arbitrary time bounds $t(x)$ from Section 9.2.2, but in a slightly weaker form. Here, instead of a generic bound $t(x)$, one gets a generic bound $t(x \log^* x)$, where \log^* is an extremely slowly growing function.

PROPOSITION 10.2.3. *Let $t : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be a time bound which is continuous, monotone increasing, and unbounded. If a function $f : I \rightarrow \mathbb{R}^+$ is bounded by t on μ -average on spheres, then f is generically $t(x \log^* x)$ relative to the asymptotic density ρ_μ .*

Proof. Suppose a function $f : I \rightarrow \mathbb{R}^+$ is bounded by t on μ -average on spheres, i.e., for some c and every n ,

$$\int_{I_n} t^{-1}(f(x)) \mu_n(x) \leq cn.$$

Let $S(f, t) = \{x \in I \mid f(x) > t(s(x) \log^* s(x))\}$. Then

$$\mu_n(S(f, t) \cap I_n) = \mu_n(\{x \in I_n \mid t^{-1}(f(x)) > s(x) \log^* x\}) \leq \frac{cn}{n \log^* n} \rightarrow \infty,$$

so $S(f, t)$ is a generic subset of I with respect to the asymptotic density ρ_μ . Hence, $t(x \log^* x)$ is a generic upper bound for f , as required. ■

10.2.3. When generically easy implies easy on average. Generic algorithms provide a powerful tool to study average case complexity of algorithmic problems. In Proposition 10.2.4 below we give a sufficient condition for an algorithmic problem to have a decision algorithm that runs within a given bound on average. The first applications of these methods appeared in [146].

Recall that a function $h : I \rightarrow \mathbb{R}^+$ is (s, μ) -balanced on a subset $K \subseteq I$ if there exists an s -upper bound $U : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ (so $h(w) \leq U(s(w))$), which is (s, μ) balanced on K , i.e.,

$$\sum_n U(n)n^{-1}\mu(K \cap I_n) < \infty.$$

PROPOSITION 10.2.4 (Generic test). *Suppose a problem \mathcal{D} is decidable in time $f(n)$ by a total decision algorithm \mathcal{B} , and generically decidable (with respect to the asymptotic density ρ_μ) in time $f_1(n)$ by a partial algorithm \mathcal{B}_1 on a generic set $K \subseteq I$. If the function $f_1^{-1}(f(n))$ is (s, μ) -balanced on $I - K$, then \mathcal{D} is decidable on μ -average in time bounded by $f_1(n)$.*

Proof. In the notation above define \mathcal{A} to be the algorithm consisting of running \mathcal{B} and \mathcal{B}_1 concurrently. Clearly, \mathcal{A} is a decision algorithm for \mathcal{D} . Observe that $T_{\mathcal{A}}(w) \leq f_1(s(w))$ if $w \in K$, and $T_{\mathcal{A}}(w) \leq f(s(w))$ otherwise. By Proposition 9.2.21, the algorithm \mathcal{A} runs within the time bound f_1 on μ -average. The result follows. ■

Now that we have demonstrated two particular applications of the Generic test, one can construct more examples in a similar fashion.

Recall that a non-negative function $f(n)$ is *subexponential* if for any $r > 1$ we have

$$\lim_{n \rightarrow \infty} \frac{f(n)}{r^n} = 0.$$

Note that this implies that for every $r > 1$,

$$\sum_{n=1}^{\infty} \frac{f(n)}{r^n} < \infty.$$

Also, $f(n)$ is *superpolynomial* if for every $k > 1$,

$$\lim_{n \rightarrow \infty} \frac{n^k}{f(n)} = 0.$$

In this case for every $k > 1$,

$$\sum_{n=1}^{\infty} \frac{n^k}{f(n)} < \infty.$$

COROLLARY 10.2.5. *Suppose a problem \mathcal{D} satisfies the conditions of Proposition 10.2.4. Then:*

- 1) *If f_1 is a polynomial, $f_1(x) = cx^m$, f is subexponential, and the residual probability function $\delta_{\bar{K}} : n \rightarrow \mu(\bar{K} \cap I_n)$ for K approaches zero exponentially fast, then \mathcal{D} has polynomial on μ -average time complexity.*
- 2) *If f_1 is exponential, $f_1(x) = a^x$, where $a > 1$, f is superexponential, $f(x) = a^{x^k}$ for some $k > 1$, and the residual probability function $\delta_{\bar{K}} : n \rightarrow \mu(\bar{K} \cap I_n)$ for K approaches zero superpolynomially fast, then \mathcal{D} has exponential on μ -average time complexity.*

Proof. To prove 1) it suffices to show that $f_1^{-1}(T_A(w))$ has an (s, μ) -balanced s -upper bound. Put $U = f_1^{-1}(f(w))$, then $f_1^{-1}(T_A(w)) \leq f_1^{-1}(f(s(w))) = U(s(w))$, so U is an s -upper bound for $f_1^{-1}(T_A(w))$. Note that $U(n) = (c^{-1}f(w))^{\frac{1}{k}}$ is subexponential. Since the residual probability function $\delta_K : n \rightarrow \mu(\bar{K} \cap I_n)$ for K approaches zero exponentially fast, this implies that there exist constants $0 < q < 1$ and $0 < d$ such that for every n ,

$$\mu(\bar{K} \cap I_n) \leq dq^n.$$

Therefore,

$$\sum_n U(n)n^{-1}\mu(\bar{K} \cap I_n) \leq d\sum_n U(n)q^n < \infty,$$

so U is (s, μ) -balanced, as claimed.

2) Similarly, if we put $U(n) = n^k$, then

$$f_1^{-1}(T_A(w)) \leq f_1^{-1}(f(s(w))) = \log_a(a^{(s(w))^k}) = s(w)^k = U(s(w)),$$

so U is an s -upper bound for $f_1^{-1}(T_A(w))$. Clearly, U is (s, μ) -balanced, since

$$\sum_n U(n)n^{-1}\mu(\bar{K} \cap I_n) \leq \sum_n n^{k-1}\mu(\bar{K} \cap I_n) < \infty,$$

as required. ■

CHAPTER 11

Generic Complexity of NP-complete Problems

In this chapter, following [98], we show that the **NP**-complete problems Subset Sum and 3-Sat have low generic complexity. Since it is well known that these problems are easy most of the time, these results confirm our expectations. The difficult instances are rare; we discuss them, too.

11.1. The linear generic time complexity of subset sum problem

There are two versions of the subset sum problem. The input in both is a sequence of natural numbers c, w_1, \dots, w_n . The decision problem is to determine if the equation $\sum x_i w_i = c$ has a solution for $x_i = 0, 1$. The optimization problem is to maximize $s = \sum x_i w_i$ subject to $s \leq c$ and $x_i = 0, 1$.

The subset sum decision problem is **NP**-complete, but the optimization problem is routinely solved in practice and gives a solution to the decision problem. It seems that difficult instances of subset sum are rare. We will show that there is a linear time algorithm which works on a generic set of inputs. For a complete discussion of the subset sum problem see the survey [153].

Our first problem here is to define a generic set. When discussing **NP**-completeness, the natural size of an instance is the number of bits in the input or something close to that. Accordingly, we let B_n be the set of inputs of length n .

Let us be more precise now. Pick an alphabet $\{0, 1, \hat{1}\}$ and consider the language L of all strings beginning with $\hat{1}$. For any string in L , a substring which begins with $\hat{1}$ and continues up to but not including the next $\hat{1}$ (or up to the end of the string) is interpreted as a binary number by taking $\hat{1} = 1$. In this way each element of L yields a sequence of binary numbers. The first number is c and the subsequent ones are the w_i 's. For example, $c = 5, w_1 = 3, w_2 = 4$ is encoded as $a01a1a00$.

The definition of a generic set is now clear: B_n , the ball of radius n , consists of all words in L of length at most n , and a subset $M \subset L$ is generic if

$$\lim_{n \rightarrow \infty} \frac{|M \cap B_n|}{|B_n|} = 1.$$

We will use the following sufficient condition for M to be generic. Let S_n be the sphere of radius n ; that is, S consists of the 3^{n-1} strings in L of length exactly n . It is straightforward to show that M is generic if

$$\lim_{n \rightarrow \infty} \frac{|M \cap S_n|}{|S_n|} = 1.$$

Our algorithm is trivial. Given input c, w_1, \dots, w_n , it checks if c is equal to some w_i . If so, the algorithm outputs “Yes” and otherwise “Don’t know”. To

complete our analysis we need to check that the set of inputs for which c matches some w_i is generic.

We will bound the fraction (or measure) of inputs in S_n for which there is no match. Let w be an input string of length n . If w contains no $\hat{1}$'s except the first one, then there is no match because there are no w_i 's. The number of w 's of this sort is 2^{n-1} and their measure in S_n is $(\frac{2}{3})^{n-1}$.

Next consider inputs whose first prefix of the form $\hat{1}(0+1)^*\hat{1}$ is of length at least $m = \lfloor (\log n)/2 \log 3 \rfloor$. The measure of this set is bounded from above by

$$\sum_{k=m}^{\infty} \left(\frac{2}{3}\right)^k = \left(\frac{2}{3}\right)^m \left(1 - \frac{2}{3}\right)^{-1} = 3 \left(\frac{2}{3}\right)^m \leq 3 \left(\frac{2}{3}\right)^{(\log n)/2 \log 3 - 1} \leq \frac{9}{4} n^{-1/5}.$$

Finally, suppose the input w has the first prefix of length k , $2 \leq k < m$. Divide k into n : $n = kq + r$, $0 \leq r < k$. Now w is a product of q subwords of length k followed by a suffix of length r . Since the first block is not duplicated, the number of possibilities for each subsequent block is $(3^k - 1)$. Thus the number of w 's of this form without a match for c is at most $2^{k-2}(3^k - 1)^{q-1}3^r \leq (3^k - 1)^q 3^r$. The measure is bounded from above by

$$\left(\frac{1}{3}\right)^{n-1} (3^k - 1)^q 3^r = 3 \left(1 - \left(\frac{1}{3}\right)^k\right)^q \leq 3 \left(1 - \left(\frac{1}{3}\right)^k\right)^{(n/k)-1} \leq 3 \left(1 - \left(\frac{1}{3}\right)^m\right)^{(n/m)-1}.$$

Now $e^{-x} \geq 1 - x$ for all x implies $\left(1 - \frac{1}{x}\right)^x \leq 1/e$. Thus

$$\left(1 - \left(\frac{1}{3}\right)^m\right)^{(n/m)-1} \leq \left(\frac{1}{e}\right)^{((n/m)-1)/3^m}.$$

But $3^m \leq 3^{(\log n)/(2 \log 3)} = \sqrt{n}$. Therefore,

$$((n/m)-1)/3^m \geq \sqrt{n}/m - 1/\sqrt{n} \geq (2 \log 3)(\sqrt{n}/\log n) - 1/\sqrt{n} \geq n^{1/3}$$

for n large enough.

We conclude that the measure in S_n of the inputs for which c does not match any w_i is at most

$$\left(\frac{2}{3}\right)^{n-1} + \frac{9}{4} n^{-1/5} + 3 \left(\frac{1}{e}\right)^{n^{1/3}} = O(n^{-1/5}).$$

11.2. A practical algorithm for the subset sum problem

We present a more practical algorithm which works well with respect to a different stratification of subset sum. Our algorithm is adapted from [52].

Suppose that for some positive integer b the weights are chosen uniformly from the interval $[1, b]$ (in Z) and c is chosen uniformly from $[1, nb]$.

ALGORITHM 11.2.1. Compute $w_1 + w_2 + \dots$ until one of the following happens.

- (1) If $w_1 + w_2 + \dots + w_j = c$, say “Yes” and halt.
- (2) If $w_1 + w_2 + \dots + w_n < c$, say “No” and halt.
- (3) If $w_1 + w_2 + \dots + w_{j-1} < c < w_1 + w_2 + \dots + w_j$, then:
 - (a) If $w_1 + w_2 + \dots + w_{j-1} + w_k = c$ for some k with $j < k \leq n$, say “Yes” and halt.
 - (b) Else say “Don’t know” and halt.

Clearly, Algorithm 11.2.1 is correct and runs in linear time. Let us estimate the probability of a “Don’t know” answer. Since c is chosen uniformly from the interval $[1, nb]$, the probability of $w_1 + w_2 + \dots + w_{j-1} < c < w_1 + w_2 + \dots + w_j$ is $(w_j - 1)/nb \leq$

$1/n$. Furthermore, the probability that there is no suitable x_k is $((b-1)/b)^{(n-j)}$. Thus the probability of “Don’t know” is at most $\sum_{j=1}^n (1/n)((b-1)/b)^{(n-j)} \leq b/n$.

Now consider the set W of all instances of the subset sum problem and let X be the subset on which Algorithm 11.2.1 returns “yes” or “no”. An instance is a tuple consisting of a constraint and a finite sequence of weights, (c, w_1, w_2, \dots) . Stratify W by defining S_n , the sphere of radius n , to be all tuples $(c, w_1, w_2, \dots, w_n)$ with $w_i \leq \sqrt{n}$ and $c \leq n^{3/2}$. In other words, $b = \sqrt{n}$. The probability distribution defined above is the equiprobable measure on S_n whence $|X \cup S_n|/|S_n| \geq 1 - b/n = 1 - 1/\sqrt{n}$. Thus W is generic.

11.3. 3-Satisfiability

3-SAT is a well-known **NP**-complete problem. An instance of 3-SAT is a finite set of clauses each consisting of the disjunction of three variables or negated variables, e.g.

$$(14) \quad \{(x_1 \vee \neg x_4 \vee x_7), (\neg x_3 \vee x_4 \vee x_6), \dots\}.$$

The problem is to decide whether or not there is a truth assignment which makes all the clauses true. The corresponding optimization problem is to find a truth assignment which makes as many as possible of the clauses true.

3-SAT has been much investigated; overviews are given in [25, Section 8] and [43]. Various ensembles of probability distributions have been proposed. A popular one is to introduce for each n a parameter m_n and consider only instances which are formed by choosing m_n clauses uniformly at random from the $8 \binom{n}{3}$ clauses composed of distinct variables. In our language and under the mild restriction that m_n is strictly increasing as a function of n , these are the instances of size n and form the sphere of radius n .

The motivation for introducing the parameter m_n is, first, that for fixed n the difficulty of an instance of 3-SAT seems to depend on the density, $c = m_n/n$, and second, in practical applications the density is often more or less constant. Extensive experiments with various algorithms have shown that as c increases from zero, the average runtime increases to a maximum and then decreases [43]. The value of the maximum depends on the particular algorithm but is independent of n [41].

If m_n is large enough ($m_n \geq C(n^{3/2})$ for a sufficiently large constant C suffices), then almost all instances are unsatisfiable; and heuristic refutation procedures suffice to show that 3-SAT $\in \textbf{AvgP}$. It does not seem to be known what happens for lower values of m_n .

By Lemma 10.1.7 of this chapter an algorithm whose time function admits an upper bound $f : I \rightarrow \mathbb{N}$ which is spherically polynomial on μ -average is generically polynomial. Thus the proof that 3-SAT $\in \textbf{AvgP}$ for m_n large enough probably also yields a proof that 3-SAT $\in \textbf{GenP}$ for m_n large enough. Below we present a slightly different approach which uses a crude refutation procedure to show 3-SAT $\in \textbf{Gen}_{\text{str}}\textbf{P}$.

We describe instances of 3-SAT as words in a language and measure size by word length. An instance of 3-SAT like 14 may be specified by the indices of the variables together with an indication of which variables are to be negated. In this notation the instance (14) becomes

$$(15) \quad 1 \vee \bar{1}00v111\#\bar{1}1 \vee 100 \vee 110\#\dots$$

where the indices are written in binary except that the leading 1 is changed to $\bar{1}$ if the variable is to be negated, and $\#$ indicates the end of a clause.

The set of instances, I , is then the set of labels of cycles from vertex α to itself in the graph Γ_1 given in Figure 11.1. In other words, I is the language accepted by the finite automaton Γ_1 with initial vertex α and terminal vertex α . The sphere I_n consists of the labels of cycles from vertex α to itself.

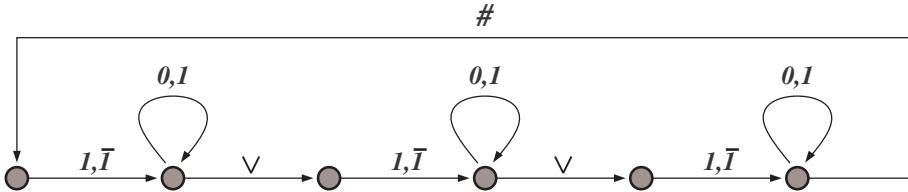


FIGURE 11.1. A finite automaton Γ accepting I . An edge with two labels stands for two edges with one label each.

Γ_1 deterministic in that no vertex has two outedges with the same label. It follows that distinct paths starting at the same vertex have distinct labels. Consequently, $|I_n|$, the size of I_n , equals the number of cycles of length n through α .

There are eight essentially different clauses with variables x_1, x_2, x_3 . As the variables in a clause may occur in any order, there are, strictly speaking, forty-eight clauses involving x_1, x_2, x_3 . If they all appear in a particular input, then that input is not satisfiable. Thus the following partial algorithm is correct.

ALGORITHM 11.3.1. Input an instance of 3-SAT
If all the clauses with variables 1, 10, 11 occur, say “No”
Else loop forever

THEOREM 11.3.2. $3\text{-SAT} \in \text{SGP}$.

Proof. As the partial algorithm runs in linear time (on its halting set), it suffices to show that the set of inputs which omit a single fixed clause with variables x_1, x_2, x_3 is strongly negligible. It will follow that the set of inputs which contain all forty-eight clauses on x_1, x_2, x_3 is strongly generic.

Let A_1 be the adjacency matrix for Γ_1 in Figure 11.1. Clearly, if the vertices of Γ_1 are partitioned into two nonempty sets, then there are edges from each set to the other. In other words, A_1 is indecomposable. By [53, Theorem I] A_1 has a positive real eigenvalue r of multiplicity one which is at least as large as the absolute value of any other eigenvalue of A_1 . In addition, increasing any entry of A_1 increases r .

The trace of A^n is equal to the sum over all vertices of the number of cycles of length n from each vertex to itself. Since every vertex is on a cycle through α , a straightforward argument shows that the size of I_n satisfies $|I_n| = \Theta(r^n)$, that is, $|I_n|$ and r^n have the same order of growth.

Let X be the set of instances in I which omit the clause $x_1 \vee x_3 \vee \neg x_2$. To complete the proof it suffices to show that I is accepted by a deterministic automaton Γ_2 such that removing an edge yields an automaton Γ_3 which accepts X ,

whose adjacency matrix A_3 is indecomposable, and to which [53, Theorem I] applies. Indeed, suppose this is so, and let A_2 be the adjacency matrix of Γ_2 . As A_2 is obtained by increasing an entry of A_3 , A_2 is also indecomposable. In addition, since Γ_2 is a deterministic automaton accepting I , its maximum real eigenvalue is r . By [53, Theorem I] the maximum real eigenvalue of A_3 is $s < r$. Thus $|X \cap I_n|/|I_n| = O((s/r)^n)$. Hence X is negligible as required.

We leave it to the reader to check that we may take Γ_2 to be the automaton in Figure 11.2. Γ_3 is obtained by removing the outedge with label $\#$ at vertex β . ■

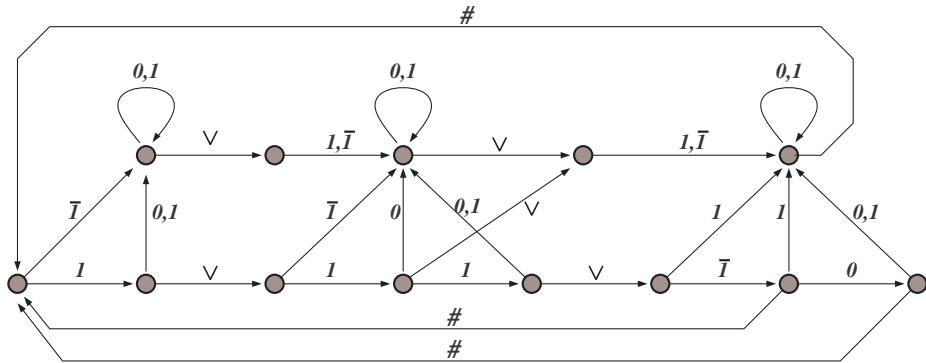


FIGURE 11.2. Another finite automaton Γ accepting I . Vertex α is the initial vertex and the single terminal vertex.

CHAPTER 12

Generic Complexity of Undecidable Problems

12.1. The halting problem

The halting problem for Turing machines is perhaps the canonical undecidable set. Nevertheless, it has been shown in [127] that there is a polynomial time algorithm deciding almost all instances of the problem.

The proof is sensitive to the particular computational model. We use the Turing machine model with a finite program directing the operation of a head reading and writing 0's and 1's while moving on a one-way infinite tape.

DEFINITION 12.1.1. The *halting problem* is the set H of programs p that halt or break when computing on input 0, on a tape initially filled with 0's.

Recall (see Section 3.1) that a Turing machine has a finite number n of states $Q_n = \{q_1, \dots, q_n\}$, with q_1 designated as the *start* state, plus a separate designated state *halt*, which is not in Q_n . We assume that all Turing machines with n states have the same set of states Q_n . A Turing machine *program* is a function

$$p : Q_n \times \{0, 1\} \rightarrow (Q_n \cup \{\text{halt}\}) \times \{0, 1\} \times \{L, R\}.$$

The transition $p(q, i) = \langle r, j, R \rangle$, for example, directs that when the head is in state q reading symbol i , it should change to state r , write symbol j , and move one cell to the right. The computation of a program proceeds by iteratively performing the instructions of such transition rules, halting when the *halt* state is reached. If the machine attempts to move left from the left-most cell, then the head falls off the tape and all computation ceases. In this case we say that the machine “breaks”.

In what follows we do not distinguish between Turing machines and Turing machine programs. Let P be a set of all Turing machine programs. By size of a program $p \in P$ we understand the number of states $s(p)$ in p . This gives a size function $S : P \rightarrow \mathbb{N}$. The sphere P_n consists of all programs with n states. Since the domain of the program has size $2n$ and the target space has size $4(n+1)$, we can easily count the number of programs in P_n :

$$\text{LEMMA 12.1.2. } |P_n| = (4(n+1))^{2n}.$$

The most natural method for measuring the size of a set of Turing machine programs is that of asymptotic density ρ_μ relative to the ensemble $\{\mu_n\}$ of the uniform distributions on spheres P_n . Thus, for a set $B \subseteq P$ of Turing machine programs

$$\rho_\mu(B) = \lim_{n \rightarrow \infty} \frac{|B \cap P_n|}{|P_n|},$$

if this limit exists. To simplify notation we omit μ in ρ_μ and write ρ .

THEOREM 12.1.3 (Generic complexity of the halting problem). *There is a set B of Turing machine programs such that:*

- 1) B has asymptotic density one.
- 2) B is polynomial time decidable.
- 3) The halting problem $H \cap B$ is polynomial time decidable.

Proof. Let B be the set of programs that on input 0 either halt before repeating a state or fall off the tape before repeating a state. To be precise, if the head falls off the tape while executing the transition $p(q, i) = \langle r, j, L \rangle$, then we do not regard the state r as having been achieved, since this step was not completed. Clearly, B is polynomial time decidable, since we need only run a program p for at most n steps, where n is the number of states in p , to determine whether or not it is in B . It is equally clear that the halting problem is polynomial time decidable for programs p in B , since again we need only simulate p for n steps to know whether it halted or fell off. What remains is to prove that this behavior occurs with asymptotic probability one, i.e., $\rho(B) = 1$. In fact, we prove a slightly stronger result that some subset B' of B is already generic in P relative to ρ . Let B' be a set of all the programs that fall off the tape before repeating a state. Notice, that b' also satisfies the conditions 2) and 3) above.

CLAIM 12.1.4. The set B' is generic, i.e., $\rho(B') = 1$.

Clearly, the theorem follows from the claim since $B' \subseteq B$. To prove the claim we need several lemmas.

LEMMA 12.1.5. *For any fixed input and fixed $k \geq 0$, the set C_k of programs not repeating states and not halting within the first k steps has asymptotic probability one.*

Proof. We show by induction on k that C_k has asymptotic density one. When $k = 0$, then all programs have the property. Suppose that the set C_k has asymptotic probability one, and consider C_{k+1} . Fix any ε , and choose n large enough so that $C_k \cap P_n$ has proportion more than $1 - \varepsilon/2$ of all programs in P_n . For a program p in $C_k \cap P_n$, consider the probability that p is in C_{k+1} . If p leads to a computation where the head has already fallen off the tape, then of course $p \in C_{k+1}$. Otherwise, the first k steps of computation by p have led to the successive states $q_{i_0}, q_{i_1}, \dots, q_{i_k}$, which have not yet repeated. The $(k+1)^{th}$ step of computation involves a transition rule $p(q_{i_k}, j_k) = (q_{i_{k+1}}, j_{k+1}, m_k)$, giving, respectively, the new state, the new bit to write on the tape and the direction to move the head. In order for p to be in C_{k+1} , it suffices that $q_{i_{k+1}}$ must not be one of the previously used states $\{q_{i_0}, q_{i_1}, \dots, q_{i_k}\}$ and not the halting state *halt*. Since there are $n - (k + 1) = n - k - 1$ many other equally likely states to choose from, the proportion of all n -state programs agreeing with p on the first k steps and satisfying the additional requirement is $\frac{n-k-1}{n}$. The proportion of all n -state programs in C_{k+1} , consequently, is at least $(1 - \varepsilon/2)(\frac{n-k-1}{n})$. Since $\frac{n-k-1}{n}$ goes to 1 as n becomes large, we may choose n large enough so that this proportion is at least $1 - \varepsilon$. Thus, C_{k+1} has asymptotic probability one, as desired. ■

To proceed with the main argument we need the following observation. Execution of a program $p \in P$ (on a fixed input on the tape) gives rise to a “walk” w_p on \mathbb{Z} , when the walker moves one step to the left or one step to the right according to the instructions of p . The walk w_p starts at 0 if the left-most cell is placed at 0.

Put $C_{k,n} = C_k \cap P_n$ and $B'_{k,n} = B' \cap C_k \cap P_n$. By construction $B'_{k,n} \subseteq C_{k,n}$. To avoid trivialities we assume that $n \geq k$. The key idea is that for the first

k steps of computation (when starting on a fixed input), the programs in $C_{k,n}$ behave statistically like a random walk with uniform probability of going left or right. The reason is that if a program lands in a totally new state q , reading some symbol i , then among the programs landing in that situation and agreeing with the computation so far, exactly half of them will opt to move left and half will move right, precisely because nothing about state q has yet been determined. Because of this, we may make use of Polya's classical result on random walks on \mathbb{Z} .

To be more precise, we term programs $p, q \in C_{n,k}$ equivalent if they induce the same k -step walk $w_{p,k}$ on \mathbb{Z} , so $p \sim q \leftrightarrow w_{p,k} = w_{q,k}$. Consider a map $W_k : C_{k,n} \rightarrow \{L, R\}^k$ such that $W_k(p) = w_{p,k}$. It is easy to see that every walk $w \in \{L, R\}^k$ can be realized as $w_{p,k}$ for some particular p (just take a program p_w that does not change the content on the tape and goes left or right according to w first k steps using new states every time), so W_k is onto. The set of pre-images $W_k^{-1}(w)$ is precisely the equivalence class $[p_w]$. For $p \in C_{k,n}$ denote $w_{p,k} = (w_{p,k}(1), \dots, w_{p,k}(k))$. To compute probabilities we need the following lemma (where $\mu_{C_{n,k}}$ is the uniform probability on $C_{k,n}$).

LEMMA 12.1.6. *For every $n \geq k \geq t+1 > 0$ and $v \in \{L, R\}^t$ the following conditional probabilities are equal:*

$$\begin{aligned} & \mu_{C_{n,k}}(p \in C_{k,n} \& w_{p,k}(t+1) = R \mid w_{p,t} = v) \\ &= \mu_{C_{n,k}}(p \in C_{k,n} \& w_{p,k}(t+1) = L \mid w_{p,t} = v). \end{aligned}$$

In particular, they both are equal to $\frac{1}{2}$.

Proof. Let $p \in C_{k,n}$ and $w_{p,t} = v$. Let

$$(16) \quad p(q_{i_1}, j_1) = (q_{i_{1+1}}, j_{1+1}, m_1), \dots, p(q_{i_t}, j_t) = (q_{i_{t+1}}, j_{t+1}, m_t)$$

be the computation by p up to the step t , so $(m_1, \dots, m_t) = v$. If $p(q_{i_{t+1}}, j_{t+1}) = (q_{i_{t+2}}, j_{t+2}, R)$ is the next transition for p then, since $q_{i_{t+1}}$ does not appear in all the previous transitions, there is a program $p' \in C_{k,n}$ that has the same t transitions (16) as p , so $w_{p',t} = w_{p,t} = v$, but p' also has the transition $p(q_{i_{t+1}}, j_{t+1}) = (q_{i_{t+2}}, j_{t+2}, L)$, thus, on its $t+1$ step p' moves to the left, meanwhile p moves to the right. This proves the lemma. ■

Lemma 12.1.6, indeed, allows one to treat the walks induced by programs from $C_{k,n}$ as simple random walks in k steps. It also shows that the equivalence classes have the same μ_n measure. Now, the famous Polya's theorem ([227]) tells one that for every $\varepsilon > 0$ there exists k such that the random simple walk on \mathbb{Z} visits -1 (when started at 0) with probability at least $1 - \varepsilon$ within first k steps. This implies the following key lemma.

LEMMA 12.1.7. *For every $\varepsilon > 0$ there exists k such that for every $n \geq k$ the fraction of programs in $C_{k,n}$ which break within the first k steps is at least $1 - \varepsilon$.*

Putting everything together, let us show that B' has asymptotic probability one. Fix any $\varepsilon > 0$, and by Lemma 12.1.7 find some large k such that for any $n \geq k$ the proportion of the subset $B'_{k,n}$ of programs in $C_{k,n}$ that break within the first k steps is at least $\sqrt{1 - \varepsilon}$, so $\mu_{C_{n,k}}(B'_{k,n}) \geq \sqrt{1 - \varepsilon}$. By Lemma 12.1.5, let n be large enough so that the fraction of programs in I_n that belong to $C_{k,n} = C_k \cap I_n$ exceeds $\sqrt{1 - \varepsilon}$, so $\mu_n(C_{k,n}) \geq \sqrt{1 - \varepsilon}$. Now,

$$\mu_n(B' \cap I_n) \geq \mu_n(B'_{k,n}) = \mu_{C_{n,k}}(B'_{k,n})\mu_n(C_{n,k}) \geq (\sqrt{1 - \varepsilon})^2 = 1 - \varepsilon.$$

So the set B' has asymptotic density one, and the theorem is proved. ■

Let us now clarify matters by untangling the two possibilities for programs in B , namely, (1) the programs that halt before repeating a state and (2) the programs that fall off the tape before repeating a state. Claim 12.1.4 shows that behavior (1) is very rare and behavior (2) occurs with asymptotic probability one. We record this in the following theorem.

THEOREM 12.1.8. *The asymptotic probability one behavior of Turing machine programs, on fixed input, is that the head falls off the tape before repeating a state.*

12.2. The Post correspondence problem

The set of inputs for the Post correspondence problem all finite sequences of pairs of words $(u_1, v_1) \dots (u_n, v_n)$, $n \geq 1$, over a fixed finite alphabet $\{a_1, \dots, a_k\}$, $k \geq 2$. The output is “Yes” if $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$ for some sequence of indices of length $m \geq 1$ and “No” otherwise. We define I_n to be the collection of inputs with n pairs of words of length between 1 and n .

It is well known that PCP is recursively unsolvable. Nevertheless, there is a trivial partial algorithm which works well enough to show that PCP is strongly generically polynomial.

ALGORITHM 12.2.1. Input an instance of the Post correspondence problem
If for all i , neither u_i nor v_i is a prefix of the other, say “No”
Else loop forever

For any solution $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$ it is clear that one of u_{i_1}, v_{i_1} is a prefix of the other. Thus our algorithm never gives a wrong answer.

THEOREM 12.2.2. *The Post correspondence problem is strongly generically polynomial; that is, it is in SGP.*

Proof. The size of I_n is $(1 + k + \dots + k^n)^{2n}$. If we restrict u_1 to be a prefix of v_k , then there are at most $n + 1$ possibilities for u_1 . Thus the number of inputs in I_n in which u_1 is a prefix of v_1 is no more than $(n + 1)(1 + k + \dots + k^n)^{2n-1}$. We conclude that the number of inputs in I_n for which some u_i is a prefix of v_i or vice-versa is at most $2n(n + 1)(1 + k + \dots + k^n)^{2n-1}$. Dividing this number by $|I_n|$ yields $\frac{2n(n+1)}{1+k+\dots+k^n}$, which approaches 0 exponentially fast as n goes to infinity. ■

12.3. Finitely presented semigroups with undecidable word problem

In 1947, Markov and Post, independently, constructed finitely presented semigroups with undecidable word problem. The key idea in both constructions was to simulate the computation of a Turing machine T in the word problem of a particular semigroup S_T , whose finite set of defining relations depends on the program of T . In 19 Tseitin [272] and in 19 Matiyasevich [186] came up with much shorter finite presentations of semigroups with undecidable word problem. In this section, following [197], we show that the word problems in the semigroups of the Markov and Post’s type are strongly generically decidable in polynomial time. It has been proven in [197] that similar results hold also for Tseitin’s and Matiyasevich’s semigroups.

To simplify the constructions it is convenient to modify slightly the standard description of transitions in Turing machines programs from Section 3.1.

Let T be a Turing machine with a set of states $Q = \{q_0, q_1, \dots, q_m\}$, with q_1 being an initial state and q_0 being a terminal state, a set of tape symbols $E = \{e_0, e_1, \dots, e_n\}$, and a program P whose transitions are written in one of the following forms:

- (1) $(q_i, e_j) \rightarrow (q_k, e_l)$;
- (2) $(q_i, e_j) \rightarrow (q_k, L)$;
- (3) $(q_i, e_j) \rightarrow (q_k, R)$.

We assume here that for every pair (q_i, e_j) , $i \neq 0$, the program P contains one and only one transition with the left-hand side (q_i, e_j) . We also assume (though it is not necessary) that T has one tape infinite in both directions.

Let T be a Turing machine. The *halting problem* for Turing machine T asks whether or not T halts on a given input.

THEOREM 12.3.1 (Turing). *There exists a Turing machine with undecidable halting problem.*

Let T be a Turing machine with a set of states Q , a set of tape symbols E , and a program P as above. The semigroup S_T is defined by a finite presentation $S_T = \langle A_T \mid R_T \rangle$ with a set of generators $A_T = E \cup Q \cup \{h\}$ (here h is a new letter not in $Q \cup E$) and a set of defining relations R_T described below:

- (i) For each instruction $(q_i, e_j) \rightarrow (q_k, e_l)$ in P the set R_T contains a relation $q_i e_j = q_k e_l$.
- (ii) For each instruction $(q_i, e_j) \rightarrow (q_k, L)$ in P the set R_T contains a set of relations $e_k q_i e_j = q_k e_k e_j$ for all $k = 0, 1, \dots, n$, and a relation $h q_i e_j = h q_i e_0 e_j$.
- (iii) For each instruction $(q_i, e_j) \rightarrow (q_k, R)$ in P the set R_T contains a set of relations $q_i e_j e_k = e_j q_k e_k$ for all $k = 0, 1, \dots, n$ and a relation $q_i e_j h = e_j q_i e_0 h$.
- (iv) $q_0 a_i = q_0, a_i q_0 = q_0$ for all $0 \leq i \leq n$ and $h q_0 h = q_0$.

Recall, that the *word problem* in a finitely presented semigroup $S = \langle A \mid R \rangle$ is a set of pairs $WP = \{(u, v) \mid u =_S v\}$. The word problem in S is *decidable* if there exists an algorithm to decide whether a given pair $(u, v) \in A^* \times A^*$ belong to WP or not.

THEOREM 12.3.2 (Markov, Post). *If a Turing machine T has undecidable halting problem then the problem to determine if a given word $w \in (A_T)^*$ is equal to q_0 in S_T is undecidable. In particular, the word problem in S_T is undecidable.*

Below we present a simple polynomial time partial algorithm that generically decides the word problem in the semigroup S_T for every Turing machine T . Notice, that the set of inputs in this case is $I = A_T^* \times A_T^*$. By the size function $s : I \rightarrow \mathbb{N}$ we understand the total length $|u| + |v|$ of the components of a pair $(u, v) \in I$. Let μ_n be the uniform distribution on the sphere $I_n = \{(u, v) \in I \mid |u| + |v| = n\}$ and ρ the asymptotic density with respect to the ensemble of the spherical distributions $\mu = \{\mu_n\}$, so for a subset $R \subseteq I$,

$$\rho(R) = \lim_{n \rightarrow \infty} \rho_n(R) = \lim_{n \rightarrow \infty} \frac{|R \cap I_n|}{|I_n|},$$

if it exists.

THEOREM 12.3.3. *Let T be a Turing machine and $S_T = \langle A_T \mid R_T \rangle$ the corresponding semigroup. Then there exists a subset $B \subseteq A_T^* \times A_T^*$ such that:*

- 1) *B is generic relative to ρ .*
- 2) *B is polynomial time decidable.*
- 3) *The word problem for pairs in B is polynomial time decidable.*

Proof. For a word $w \in A_T^*$ we denote by $\delta_q(w)$ the total number of symbols from Q in w . Observe, that for every relation $r = s$ from R_T one has $\delta_q(r) = \delta_q(s)$. Therefore, if two words $u, v \in A_T^*$ are equal in S_T then $\delta_q(u) = \delta_q(v)$. This gives a simple test of non-equal elements in S_T :

$$\forall u, v \in A_T^* (\delta_q(u) \neq \delta_q(v) \longrightarrow u \neq_{S_T} v).$$

Our generic algorithm \mathcal{A} is just an implementation of this test: given two words $u, v \in A_T^*$ the algorithm \mathcal{A} computes $\delta_q(u)$ and $\delta_q(v)$ and says “No” if $\delta_q(u) \neq \delta_q(v)$, otherwise it says “Do not know” (or goes into an infinite loop, if one prefers no answers of the type “Do not know”). Clearly, \mathcal{A} is a polynomial time partial decision algorithm for the word problem in S_T . Put

$$B = \{(u, v) \in A_T^* \times A_T^* \mid \delta_q(u) \neq \delta_q(v)\}.$$

Obviously, B is polynomial time decidable and \mathcal{A} gives the correct answer “No” for every pair of elements $(u, v) \in B$. This shows that B satisfies the conditions 2) and 3) from the statement of the theorem. To finish the proof one needs to show that the set B is generic in $A_T^* \times A_T^*$ with respect to the standard spherical asymptotic density ρ . This has been shown in [197], the proof is rather technical, so we refer the reader there for details.

The final remark on the word problem in S_T concerns the complexity of the problem: decide if a given $w \in A_T^*$ is equal to q_0 in S_T or not (symbolically $w = q_0?$). According to the Markov-Post Theorem 12.3.2 the principal source (in fact, the only known one) of undecidability of the word problem in S_T comes from the problem whether $w = q_0$ or not (and, of course, from the undecidability of the halting problem). However, we show below that the problem $w = q_0?$ is strongly generically decidable. This indicates, perhaps, that the word problem in S_T maybe strongly generically decidable, but the question is open still.

LEMMA 12.3.4. *Let $\delta \in \mathbb{N}$ and $S = \{w \in A_T^* \mid \delta_q(w) \neq \delta\}$. Then S is strongly generic in A_T^* .*

Proof. We fix $\delta \in \mathbb{N}$ and $i \in \mathbb{N}$ such that $i \neq \delta$. Define $S_i = S \cap B_i$. So we need to show that

$$f_i = \frac{|S_i|}{|B_i|} \rightarrow 1.$$

Lets take a closer look at elements in B_i . They are strings of length i in alphabet X . It is clear that $|B_i| = |X|^i$. Elements of S_i are those from B_i with the number of q -symbols different from δ . Notice that $|X| = n + m + 3$. Now, if we denote by p the fraction $\frac{m+1}{m+n+3}$ of the number of q symbols to the number of all symbols then $|S_i| = |B_i| - \binom{i}{\delta} p^\delta (1-p)^{(i-\delta)}$. Thus

$$\begin{aligned} f_i &= \frac{|S_i|}{|B_i|} = \frac{|X|^i - \binom{i}{\delta} p^\delta (1-p)^{(i-\delta)}}{|X|^i} \\ &= 1 - \frac{\binom{i}{\delta} p^\delta (1-p)^{(i-\delta)}}{|X|^i}. \end{aligned}$$

For a fixed δ and i tending to infinity in the numerator we have a constant value p^δ , $\binom{i}{\delta} = \frac{i(i-1)\dots(i-\delta+1)}{\delta!}$ is a polynomial of power δ and $(1-p)^{(i-\delta)} \rightarrow 0$ as $i \rightarrow \infty$. Therefore, since $\frac{1}{|X|^i} \rightarrow 0$ exponentially fast, we have exponential convergence $f_i \rightarrow 1$. \blacksquare

PROPOSITION 12.3.5. *Let $w \in X^*$. Then the equivalence to q_0 is strongly generically linear.*

Proof. Follows from Lemma 12.3.4 and the fact that the complexity of computing δ_q is linear. \blacksquare

CHAPTER 13

Strongly, Super, and Absolutely Undecidable Problems

In this chapter, which is based on joint results with Rybalov [191], we discuss algorithmic complexity of undecidable problems. We do not study them as in abstract recursion theory, say, trying to place them in the upper echelons of the hierarchy of Turing degrees. To the contrary, we approach them from a very practical computational view-point; we study their algorithmic behavior on “most” or “typical” inputs. This approach originated in asymptotic and computational algebra, and is called now “generic complexity” (see, [13, 29, 146, 147]). Generic complexity allows one to study naturally the computational behavior of undecidable problems — impossible task in the worst-case or the average case complexities. In fact, the generic complexity provides a unifying framework for studying, and comparing, computational complexity of decidable and undecidable problems. It turned out, for example, that the famous halting problem for Turing machines (with a one-way infinite tape) is easily decidable on most inputs — on the so-called “generic” sets of inputs [127]. This result generated a thorough study of several other undecidable problems, in particular, in groups [30] and semigroups [197]. Again, it has been shown that they are easy on some generic sets of inputs. Here we discuss undecidable problems that are still undecidable on arbitrary generic sets. Moreover, we give a method on how to amplify the undecidability of a given problem onto generic sets. Finally, we introduce “absolutely” undecidable problems, whose restrictions on arbitrary non-negligible sets are still undecidable. To describe these results we need to recall a few definitions from generic complexity.

A partial decision algorithm \mathcal{A} for \mathcal{D} *generically solves* the problem \mathcal{D} if the halting set $H_{\mathcal{A}}$ of \mathcal{A} is generic in I with respect to the spherical asymptotic density ρ . If such a partial decision algorithm \mathcal{A} exists we say that \mathcal{D} is *generically decidable*. Similarly, \mathcal{D} is *strongly generically decidable* if there exists a partial decision algorithm \mathcal{A} with a strongly generic halting set $H_{\mathcal{A}}$.

As we have mentioned above the standard halting problem for Turing machines (with a one-way infinite tape), being undecidable, is nevertheless generically decidable. To study undecidable problems which are undecidable on every “large” set of inputs we need the following terminology, which may seem, at first glance, a bit counterintuitive.

We say that an undecidable problem \mathcal{D} is *strongly undecidable* if it is undecidable on every strongly generic relative to ρ subset of inputs from I . More precisely, in this case for every strongly generic subset $I' \subseteq I$ the restriction of \mathcal{D} onto I' is still undecidable. Among strongly undecidable problems there are even “more undecidable” ones — we say \mathcal{D} is *super-undecidable* if its undecidable on any generic

subset of I . Furthermore, \mathcal{D} is termed *absolutely undecidable* if it is undecidable on every non-negligible subset from I .

In Section 13.1 we introduce the required terminology for the halting problem (HP) for Turing machines and indicate why HP is generically decidable when the tape is one-ended (for a complete proof see [127]).

In Section 13.2, following [241], we show that the halting problem is, in fact, strongly undecidable. Then we prove a “strongly undecidable” analog of the Rice’s theorem, which provides plenty of examples of strongly undecidable problems. We would like to emphasize here that these results do not depend on a particular model of Turing machines.

In Section 13.3 we discuss *generic amplification* — a method that allows one to construct a strongly (super-strongly) undecidable problem from a given undecidable problem, thus amplifying the hardness of the initial problem. The main tool here is termed “cloning”. A *cloning* from a set I into a set J is a function $C : I \rightarrow P(J)$ from I into the set of all subsets $P(J)$ of J such that

$$\forall x, y \in I (x \neq y \rightarrow C(x) \cap C(y) = \emptyset).$$

A *clone* $C(S)$ of a subset $S \subseteq I$ is just union of clones of members of S , so $C(S) = \bigcup_{x \in S} C(x)$. Similarly, if $\mathcal{D} = (L, I)$ is a decision problem in I then a decision problem $C(\mathcal{D}) = (C(L), J)$ in J is called the *clone* of \mathcal{D} in J relative to C .

A cloning C from I to J is *effective* if there is an algorithm that for every $x \in I$ computes an effective enumeration of all elements in the clone $C(x)$, and it is *non-negligible*, (*non-strongly-negligible*), if for every $x \in I$ the clone $C(x)$ is non-negligible (non-strongly-negligible) in J . Theorem 13.3.3 is the main technical result of the chapter. It states that if C is an effective non-strongly-negligible (non-negligible) cloning from I to J and $\mathcal{D} = (L, I)$ is undecidable problem in I then it is clone $C(\mathcal{D}) = (C(L), J)$ is strongly undecidable (super-undecidable) problem in J . In the same section we construct several effective non-negligible clonings thus providing plenty of examples of super-undecidable problems. Although these clonings are easy to construct, they often change the nature of the original decision problem, for example, the cloning of the word problem in a given group, or the decidability problem of a first-order theory, becomes a membership problem for some obscure language in a binary alphabet. In the rest of this chapter we construct some particular clonings that preserve the nature of the original problem.

There are several famous finitely presented semigroups with undecidable word problem (see, for example, [184, 228, 272, 186], or a survey [1]). However, it has been shown in [197] that their Word Problems have polynomial time generic case complexity. In Section 13.4 we show how one can amplify the generic complexity of finitely presented semigroups with undecidable word problem. Namely, for a finitely presented semigroup \mathfrak{S} we construct a semigroup \mathfrak{S}_x whose word problem is an effective non-negligible clone of the word problem of \mathfrak{S} . It follows that if the word problem in \mathfrak{S} is undecidable then the word problem in \mathfrak{S}_x is super-undecidable.

Tseitin semigroup \mathfrak{T} has a presentation with 5 generators, 7 relations and undecidable word problem [272]. In this case the semigroup \mathfrak{T}_x with super-undecidable word problem has 6 generators and 13 relators whose total length is equal to 49.

We would like to mention here that we do not know any examples of finitely presented groups with super-undecidable word problem.

Finally, in Section 13.5 we study problems which are absolutely undecidable, i.e., undecidable one every non-negligible subset of inputs. Let $I = \{0, 1\}^*$ be the

set of all words in the binary alphabet. Recall that a subset $S \subseteq I$ is called *immune* if S is infinite and does not contain any infinite r.e. subset of I . We show first that there exists a generic immune subset of I , and prove then that every generic immune subset of I is absolutely undecidable. This provides a host of examples of absolutely undecidable problems. However, the question of whether or not there exist finitely presented semigroups (or groups, or other natural structures) with absolutely undecidable word problem — remains open.

13.1. Halting problem for Turing machines

The halting problem for Turing machines is one of the basic undecidable problems. In this section we introduce some necessary definitions and results that are used later.

Recall that a Turing machine has a finite number n of states $Q_n = \{q_1, \dots, q_n\}$, with q_1 designated as the *start* state, plus a separate designated state q_0 , which is not in Q_n . We assume that all Turing machines with n states have the same set of states Q_n . A Turing machine *program* is a function

$$p : Q_n \times \{0, 1\} \rightarrow (Q_n \cup \{q_0\}) \times \{0, 1\} \times \{L, R\}.$$

The transition $p(q, i) = \langle r, j, R \rangle$, for example, directs that when the head is in state q reading symbol i , it should change to state r , write symbol j , and move one cell to the right. The computation of a program proceeds by iteratively performing the instructions of such transition rules, halting when the final state q_0 is reached.

If the machine operates on a one-way infinite tape and attempts to move left from the left-most cell, then the head falls off the tape and all computation ceases. In this case we say that the machine “breaks”.

If a particular model of Turing machines is fixed (one-way infinite tape, two tapes, etc.) then we do not distinguish between Turing machines and Turing machine programs.

DEFINITION 13.1.1. The *Empty Tape Halting Problem* is the set H of programs p of Turing machines that halt or break when computing on a tape initially filled with 0’s.

Let P be a set of all Turing machine programs. By size of a program $p \in P$ we understand the number of states $s(p)$ in p . This gives a size function $s : P \rightarrow \mathbb{N}$. The sphere P_n consists of all programs with n states. Since the domain of the program has size $2n$ and the target space has size $4(n + 1)$, one can easily count the number of programs in P_n :

$$\text{LEMMA 13.1.2. } |P_n| = (4(n + 1))^{2n}.$$

The most natural method for measuring the size of a set of Turing machine programs is that of asymptotic density ρ relative to the ensemble $\{\mu_n\}$ of the uniform distributions on spheres P_n . Thus, for a set $B \subseteq P$ of Turing machine programs

$$\rho(B) = \lim_{n \rightarrow \infty} \frac{|B \cap P_n|}{|P_n|},$$

if this limit exists. We define *generic* and *strongly generic* sets of programs relative to the asymptotic density ρ .

The following result was obtained in [127], it holds for the model of Turing machines with a one-way infinite tape. The proof is sensitive to this particular

computational model. The question whether the result holds for arbitrary model of Turing machines is open.

THEOREM 13.1.3 (Generic decidability of the Empty Tape Halting Problem). *There is a set B of Turing machine programs such that*

- 1) B is generic.
- 2) B is polynomial time decidable.
- 3) The restriction $H \cap B$ of the empty tape halting problem H on B is polynomial time decidable.

13.2. Strongly undecidable problems

In this section we prove that some classical undecidable problems are, in fact, strongly undecidable, i.e., they are not decidable on any strongly generic subset of inputs. We show first (following [241]) that the classical halting problem is strongly undecidable. Then we prove the corresponding analog of Rice's theorem, which provides plenty of examples of strongly undecidable problems. We would like to emphasize here that these results do not depend on a model of Turing machines.

13.2.1. The halting problem is strongly undecidable. In this section, following Rybalov [241], we show that the halting problem is strongly undecidable, i.e., it is undecidable on every strongly generic set of inputs. The result holds for every model of Turing machines and (as we shall see in Section 13.2.2) every variation of the halting problem.

We refer to Section 13.1 for our description of Turing machines, except for the tape — it is arbitrary now.

Let δ be an effective coding of all Turing machines by strings in the alphabet $\{1\}$, so given $\delta(M)$ one can recover M and given a machine M one can effectively compute $\delta(M)$. We also assume that for a string $x \in \{1\}^*$ one can effectively determine if $x = \delta(M)$ for some Turing machine M or not.

Below we consider the classical version of the halting problem.

DEFINITION 13.2.1. (The classical halting problem) The halting problem is the set HP of programs p of Turing machines that halt on the input $\delta(p)$.

By definition HP is decidable on a strongly generic subset $S \subseteq P$ if there is a partial computable function $f : \{1\}^* \rightarrow \{0, 1\}$ such that $S \subseteq \text{Dom}(f)$ and if $f(\delta(M)) = 1$ then M halts on $\delta(M)$, and if $f(\delta(M)) = 0$ then M does not halt on $\delta(M)$, and also $f(x)$ is undefined if $x \neq \delta(M)$ for any Turing machine M . In this case, we say that f is a strongly generic decision function for HP . In particular, it follows that the domain $\text{Dom}(f)$ is a strongly generic recursively enumerable set on which HP is decidable.

Let M be a Turing machine with k non-final states $\{q_1, \dots, q_k\}$. For a given $n \geq k$ one can construct a new machine M^* on n states with the following program:

$$\begin{aligned} & 2k \text{ fixed instructions of } M \quad \left\{ \begin{array}{l} (q_1, 0) \rightarrow \dots, \\ \dots \\ (q_k, 1) \rightarrow \dots, \end{array} \right. \\ & \text{arbitrary } 2(n - k) \text{ instructions} \quad \left\{ \begin{array}{l} (q_{k+1}, 0) \rightarrow \dots, \\ \dots \\ (q_n, 1) \rightarrow \dots. \end{array} \right. \end{aligned}$$

It is easy to see that M^* computes the same function as M does, because the new states are not attainable from the states of M , so M^* never executes any of the new instructions. Denote by $C(M)$ the set of all Turing machines M^* described above.

LEMMA 13.2.2. *For any Turing machine M the set $C(M)$ is not strongly negligible.*

Proof. Let M be a Turing machine with k states. It is easy to see that the number of Turing machines M^* from $C(M)$ of size n is $(4(n+1))^{2(n-k)}$. By Lemma 13.1.2 $|P_n| = (4(n+1))^{2n}$, so

$$\frac{|C(M) \cap P_n|}{|P_n|} = \frac{(4(n+1))^{2(n-k)}}{(4(n+1))^{2n}} = \frac{1}{(4(n+1))^{2k}},$$

hence $C(M)$ is not strongly negligible. ■

THEOREM 13.2.3. *The halting problem HP is strongly undecidable, i.e., it is undecidable on every strongly generic subset of programs.*

Proof. We follow here the classical proof of undecidability of the halting problem (see, for example, [44]). Suppose, to the contrary, that there exists a strongly generic set S on which HP is decidable. Then there exists a partial decision function $f : \{1\}^* \rightarrow \{0, 1\}$ for HP such that $S \subseteq \text{Dom}(f)$. We may assume that $S = \text{Dom}(f)$. It is easy to see then that the following function is partial computable

$$h(x) = \begin{cases} \text{undefined}, & \text{if } f(x) = 1, \text{ or } f(x) \text{ is undefined,} \\ 1, & \text{if } f(x) = 0. \end{cases}$$

Denote by M_h a Turing machine that computes h . Notice, that the set $P - S$ is strongly negligible, so by Lemma 13.2.2 the set $C(M_h)$ is not a subset of $P - S$, hence, there is a machine M_h^* from $S \cap C(M_h)$ computing h . Now let's look at the result of computation of M_h^* on the input $\delta(M_h^*)$. Observe, first, that $f(\delta(M_h^*))$ is defined since $M_h^* \in S$. If M_h^* halts on $\delta(M_h^*)$ then $f(\delta(M_h^*)) = 1$, hence $h(\delta(M_h^*))$ is undefined, so M_h^* does not halt on $\delta(M_h^*)$ — contradiction. If M_h^* does not halt on $\delta(M_h^*)$ then $f(\delta(M_h^*)) = 0$, so $h(\delta(M_h^*)) = 1$, which implies that M_h^* halts on $\delta(M_h^*)$ — contradiction. This shows that such S do not exist, as claimed. ■

13.2.2. Strongly undecidable analog of Rice's theorem. The famous Rice's theorem states that if \mathcal{F} is a proper class of partial computable functions then the problem whether or not a given Turing machine computes a function from \mathcal{F} is undecidable. This gives a host of examples of undecidable problems. We show below that these problems are, in fact, strongly undecidable. This result holds for any model of Turing machines.

Let \mathcal{C} be a class of all partial computable functions of the type $f : \{1\}^* \rightarrow \{0, 1\}$. A subclass \mathcal{F} of \mathcal{C} is proper if $\mathcal{F} \neq \emptyset$ and $\mathcal{F} \neq \mathcal{C}$. Below we follow notation from Sections 13.1 and 13.2.1.

THEOREM 13.2.4. *Let \mathcal{F} be a proper class of partial computable functions. Then the problem whether or not a given Turing machine computes a function from \mathcal{F} is strongly undecidable.*

Proof. Let $S \subseteq P$ be a strongly generic set of Turing machines and \mathcal{A} a partial decision algorithm on P , such that the halting set of \mathcal{A} is S , and for every $M \in S$ \mathcal{A} correctly decides whether the machine M computes a function from \mathcal{F} or not. Fix

an arbitrary function $g \in \mathcal{F}$ and a Turing machine G that computes g . Without loss of generality we may assume that $\emptyset \notin \mathcal{F}$.

Now for every Turing machine M we define a Turing machine M_G such that given an input $x \in \{1\}^*$ the machine M_G prints $\delta(M)$, then starts M on $\delta(M)$ and if M halts on $\delta(M)$, then M_G starts G on x . If G halts on x then M_G outputs this result as the value of M_G on x , in all other cases M_G does not halt on x . Notice, that by construction the following claim holds.

CLAIM 13.2.5. A Turing machine M halts on $\delta(M)$ if and only if the Turing machine M_G computes a function from \mathcal{F} .

Proof. Indeed, if M halts on $\delta(M)$ then M_G computes the function $g \in \mathcal{F}$. On the other hand, if M does not halt on $\delta(M)$ then M_G never halts, so it computes the empty function \emptyset which belongs to $P - \mathcal{F}$. ■

Now, for a given Turing machine M consider the set $C(M_G)$ of Turing machines M_G^* defined in Lemma 13.2.2. Recall, that every machine from $C(M_G)$ computes the same function as the machine M_G . In particular, a machine $M_G^* \in C(M_G)$ computes a function from \mathcal{F} if and only if the machine M halts on $\delta(M)$.

Notice, that by construction the set $C(M_G)$ is effectively enumerable. Now, we design a total algorithm \mathcal{B} that decides the halting problem HP for Turing machines. Given a Turing machine M the algorithm \mathcal{B} enumerates effectively all the machines from the set $C(M_G)$ and starts the algorithm \mathcal{A} on each one of them. Since the set $C(M_G)$ is not strongly negligible the intersection $S \cap C(M_G)$ is not empty. Therefore, the algorithm \mathcal{A} will eventually stop on some machine $M_G^* \in C(M_G)$ and tell whether M_G^* computes a function from \mathcal{F} or it does not. Since M_G^* computes a function from \mathcal{F} if and only if M halts on $\delta(M)$, this solves the halting problem for M , as claimed. This contradicts the undecidability of the halting problems and proves the result. ■

13.3. Generic amplification of undecidable problems

In this section we discuss generic amplification — a method that allows one to construct a strongly (super-strongly) undecidable problem from a given undecidable problem, thus amplifying the hardness of the problem to an arbitrary strongly generic (generic) set. The main tool of the generic amplification method is “cloning”, we describe it below.

Let I and J be sets. A *cloning* of I in J is a function $C : I \rightarrow P(J)$ from I into the set of all subsets $P(J)$ of J such that

$$\forall x, y \in I (x \neq y \rightarrow C(x) \cap C(y) = \emptyset).$$

For a subset $S \subseteq I$ its *clone* $C(S)$ is defined as union of clones of elements in S :

$$C(S) = \bigcup_{x \in S} C(x).$$

Similarly, if $\mathcal{D} = (L, I)$ is a decision problem in I then a decision problem $C(\mathcal{D}) = (C(L), J)$ in J is called the *clone* of \mathcal{D} in J relative to C . Since $C(x) \cap C(y) = \emptyset$ for $x \neq y$ one has the following fundamental property of cloned problems:

$$(17) \quad x \in L \iff C(x) \subseteq C(L) \iff C(x) \cap C(L) \neq \emptyset$$

We say that cloning C from I to J is *effective* if there is an algorithm $E(x, i)$ that for every $x \in I$ computes an effective enumeration of the clone $C(x)$, i.e.,

$$(18) \quad C(x) = \{E(x, 0), E(x, 1), \dots\}.$$

Among effective clonings one can consider *polynomial*, *super-polynomial*, *exponential* time clonings, or, more generally, effective clonings with a given time bound.

One can view an effective cloning C from I to J as a total computable function $E : I \times \mathbb{N} \rightarrow J$ such that $C(x) \cap C(y) = \emptyset$ for $x \neq y$ where the clone $C(x)$ of x is defined by (18).

To amplify a worst-case hardness of a given problem \mathcal{D} in I into a generically hard distributional problem $C(\mathcal{D})$ in J one needs to have a notion of genericity of subsets of J . To this end, we assume that the set J is equipped with either a probability distribution μ , or an ensemble of spherical (volume) distributions $\mu = \{\mu_n\}$ with respect to a fixed size function $s : J \rightarrow \mathbb{N}$. In both events the notion of a generic subset of J is defined (either with respect to μ or with respect to the asymptotic density relative to $\{\mu_n\}$).

A cloning $C : I \rightarrow P(J)$ is called *non-negligible*, (*non-strongly-negligible*, etc.), if for every $x \in I$ the clone $C(x)$ is non-negligible (non-strongly-negligible, etc.) in J .

EXAMPLE 13.3.1. Let $I = J = \{0, 1\}^*$. Define a function $E : I \times \mathbb{N} \rightarrow I$ by

$$E(a_1 \dots a_{n-1} a_n, i) = a_1 0 \dots a_{n-1} 0 a_n 1 \text{bin}(i),$$

where $a_k \in \{0, 1\}$ and $\text{bin}(i)$ is the binary expression of the natural number i .

We claim that E gives rise to a polynomial time computable non-negligible cloning from I to I . Indeed, in this case, if $x = a_1 \dots a_{n-1} a_n$, then, according to (18),

$$C(x) = \{a_1 0 \dots a_{n-1} 0 a_n 1 \text{bin}(i) \mid i \in \mathbb{N}\},$$

so $C(x) \cap C(y) = \emptyset$ for $x \neq y$, hence C is cloning from I to I . Clearly, C is effective, moreover, E is polynomial time computable. Observe, that for the spherical uniform distribution μ_n on the sphere $I_n = \{w \in I \mid |w| = n\}$ one has for $n \geq 2|x| + 1$:

$$\mu_n(C(x) \cap I_n) = \frac{2^{n-2|x|-1}}{2^n} = \frac{1}{2^{2|x|+1}} > 0.$$

Therefore,

$$\rho(C(x)) = \frac{1}{2^{2|x|}} > 0$$

and $C(x)$ is non-negligible for any x .

EXAMPLE 13.3.2 (Cloning into a two-letter alphabet). Here for an arbitrary finite alphabet A we construct an effective non-negligible cloning C_A from A^* into $\{0, 1\}^*$. Fix an injective map $\alpha : A \rightarrow \{0, 1\}^*$ such that for any $a \in A$ its image $\alpha(a)$ has length $\lceil \log_2 |A| \rceil + 1$. The map α gives rise to a homomorphism of monoids $\alpha^* : A^* \rightarrow \{0, 1\}^*$. Now it is easy to see that the composition

$$C_A = C \circ \alpha^* : A^* \rightarrow P(\{0, 1\}^*),$$

where C is the cloning constructed in Example 13.3.1, is a non-negligible effective cloning function.

THEOREM 13.3.3. *Let I, J be sets and $C : I \rightarrow P(J)$ an effective cloning. Then for every problem $\mathcal{D} = (L, I)$ in I the following holds:*

- 1) if C is non-negligible and the clone problem $C(\mathcal{D})$ is generically decidable in J then \mathcal{D} is decidable in I .
- 2) if C is non-strongly-negligible and the clone $C(\mathcal{D})$ is strongly generically decidable in J then \mathcal{D} is decidable in I .

Proof. Let $\mathcal{D} = (L, I)$ be a decision problem in I and $C(\mathcal{D}) = (C(L), J)$ is its C -clone in J . For any partial decision algorithm \mathcal{A} for $C(\mathcal{D})$ in J we design a total decision algorithm \mathcal{A}' for \mathcal{D} in I as follows. In the notation above the algorithm \mathcal{A}' works on input $x \in I$ as follows.

- (1) For $i = 0, 1, 2, \dots$
- (2) Compute the element $E(x, i)$ and start the algorithm \mathcal{A} on $E(x, i)$.
- (3) If for some i \mathcal{A} halts on $E(x, i)$ then \mathcal{A}' outputs the value $\mathcal{A}(E(x, i))$.

Since

$$x \in L \leftrightarrow C(x) \cap C(L) \neq \emptyset$$

the algorithm \mathcal{A}' is a correct partial decision algorithm for \mathcal{D} . To see that \mathcal{A}' is total it suffices to notice that for any $x \in I$ the clone $C(x)$ intersects non-trivially with the halting set $H_{\mathcal{A}}$ of the algorithm \mathcal{A} . Indeed, under the assumptions of the theorem the clone $C(x)$ is not a subset of the complement of $H_{\mathcal{A}}$ in J because if \mathcal{A} is generic (as in 1) then $H_{\mathcal{A}}$ is negligible, but $C(x)$ is non-negligible; similarly, if \mathcal{A} is strongly generic (as in 2) then $H_{\mathcal{A}}$ is strongly negligible, but $C(x)$ is non-strongly-negligible. This proves the theorem. ■

COROLLARY 13.3.4. *The following hold:*

- 1) Let $I = \{0, 1\}^*$ and C the cloning from Example 13.3.1 Then for every undecidable problem $\mathcal{D} = (L, I)$ its clone problem $C(\mathcal{D}) = (C(L), I)$ is super-undecidable.
- 2) Let A be a finite alphabet, $I = A^*$, and C_A the cloning from Example 13.3.2. Then for every undecidable problem $\mathcal{D} = (L, I)$ its clone problem $C(\mathcal{D}) = (C(L), I)$ is super-undecidable.

We finish this section with a discussion on possible relaxations of the notion of cloning. We start with the following example.

EXAMPLE 13.3.5. Let P be a set of all Turing machine programs. For a program $p \in P$ denote by $C(p)$ the set of programs of Turing machines introduced in Lemma 13.2.2 from Section 13.2.1. Then $C(p)$ is not strongly negligible for every p and there is a total computable function $E(x, i)$ satisfying (18), but $p \rightarrow C(p)$ is not a cloning since $C(x)$ and $C(y)$ may intersect non-trivially for $x \neq y$. Notice, however, that the argument in the proof of Theorem 13.2.4 still works in this case, since the fundamental property (17) holds here.

The Example 13.3.5 shows that to prove Theorem 13.2.4 one needs only a function that non-strongly negligibly m -reduces the halting problem HP for Turing machines to the problem of whether a given Turing machine program computes a function from the proper class \mathcal{F} or not. However, the theory of generic m -reductions is not developed yet, and it remains to be seen how far such a development can go along the classical lines.

13.4. Semigroups with super-undecidable word problems

There are several famous finitely presented semigroups with undecidable word problem (see, for example, [184, 272, 186], or a survey [1]). However, it has been shown in [197] that their word problems have polynomial time generic case complexity. The main reason for this phenomenon is that to interpret a hard problem (say, the halting problem for Turing machines) one brings into the finite presentation of a semigroup a lot of “garbage” (extra generators and relators that serve to simulate the Turing machine computation) that makes the word problem easy on most inputs. In this section we show how one can amplify the generic complexity of finitely presented semigroups with undecidable word problem. Namely, we describe a general method to construct finitely presented semigroups with super-undecidable word problems. More precisely, for a finitely presented semigroup \mathfrak{S} we construct a semigroup \mathfrak{S}_x whose word problem is an effective non-negligible clone of the word problem of \mathfrak{S} .

Let

$$\mathfrak{S} = \langle a_1, \dots, a_n \mid r_1 = s_1, \dots, r_k = s_k \rangle$$

be a finitely presented semigroup with a set of generators $A = \{a_1, \dots, a_n\}$ and a set of defining relations $R = \{r_1 = s_1, \dots, r_k = s_k\}$.

Denote by $WP_{\mathfrak{S}}$ the word problem in the semigroup \mathfrak{S} , so

$$WP_{\mathfrak{S}} = \{(u, v) \in A^* \times A^* \mid u = v \text{ in } \mathfrak{S}\}.$$

For a letter $x \notin A$ put

$$\mathfrak{S}_x = \langle A, x \mid R, x = xa_1, \dots, x = xa_n, x = xx \rangle.$$

Then \mathfrak{S}_x is also a finitely presented semigroup. Denote $A_x = A \cup \{x\}$.

LEMMA 13.4.1. *For any $w_1, w_2 \in A^*$ and $v_1, v_2 \in A_x^*$ the following hold:*

- 1) $w_1 = w_2$ in $\mathfrak{S} \Leftrightarrow w_1 = w_2$ in \mathfrak{S}_x ,
- 2) $w_1 = w_2$ in $\mathfrak{S} \Leftrightarrow w_1 xv_1 = w_2 xv_2$ in \mathfrak{S}_x .

Proof. In the semigroup \mathfrak{S}_x only rules from R are applicable to words that do not contain x . This shows 1).

On the other hand, if a word $w \in A_x^*$ contains a letter x , then, using the rules $x = xa_i$ and $x = xx$ from \mathfrak{S}_x , one can delete all the letters in w to the right of a given occurrence of x in w . This shows that for any $w_1, w_2 \in A^*$ and $v_1, v_2 \in A_x^*$ one has

$$w_1 xv_1 = w_2 xv_2 \text{ in } \mathfrak{S}_x \Leftrightarrow w_1 x = w_2 x \text{ in } \mathfrak{S}_x.$$

Now, the only rules that are applicable in \mathfrak{S}_x to the words $w_1 x$, $w_2 x$ are the rules that belong to R . So

$$w_1 x = w_2 x \text{ in } \mathfrak{S}_x \Leftrightarrow w_1 = w_2 \text{ in } \mathfrak{S}.$$

This shows 2). ■

COROLLARY 13.4.2. *The canonical embedding $A \rightarrow A_x$ extends to an embedding of semigroups $\mathfrak{S} \rightarrow \mathfrak{S}_x$.*

LEMMA 13.4.3. *Let $I = A^* \times A^*$ and $J = A_x^* \times A_x^*$. For $(u, v) \in I$ put*

$$C(u, v) = \{(uxp, vxq) \mid p, q \in A_x^*\}.$$

Then C is an effective non-negligible cloning.

Proof. To show that C is a cloning it suffices to notice that

$$C(u, v) = C(u_1, v_1) \iff u = u_1 \text{ and } v = v_1,$$

which is obvious.

We claim that C is an effective cloning. To see this, let

$$\tau : \mathbb{N} \rightarrow A_x^* \times A_x^*,$$

be a computable bijective enumeration of all pairs of words in alphabet A_x . Notice, that for every $i \in \mathbb{N}$,

$$\tau(i) = (\tau_1(i), \tau_2(i))$$

for some uniquely defined computable functions $\tau_1 : \mathbb{N} \rightarrow A_x^*$, $\tau_2 : \mathbb{N} \rightarrow A_x^*$. Now, a function $E : A^* \times A^* \times \mathbb{N} \rightarrow A_x^* \times A_x^*$ defined as

$$E(u, v, i) = (ux\tau_1(i), vx\tau_2(i))$$

gives an effective enumeration of the clones $C(u, v)$, $(u, v) \in I$.

We claim now that C is a non-negligible cloning. By definition the size of a pair of $(u, v) \in J$ is equal to the sum of their lengths, so the sphere J_n of radius n in J is the following set

$$J_n = \{(p, q) \in A_x^* \times A_x^* \mid |p| + |q| = n\}.$$

Therefore,

$$|J_n| = \sum_{i=0}^n |A_x|^i |A_x|^{n-i} = (n+1)|A_x|^n.$$

Similarly,

$$|C(u, v) \cap J_n| = \sum_{i=0}^{n-|u|-|v|-2} |A_x|^i |A_x|^{n-i} = (n - |u| - |v| - 1)|A_x|^{n-|u|-|v|-2}.$$

Hence

$$\mu_n(C(u, v) \cap J_n) = \frac{(n - |u| - |v| - 1)|A_x|^{n-|u|-|v|-2}}{(n+1)|A_x|^n} \rightarrow \frac{1}{|A_x|^{|u|+|v|+2}} > 0.$$

This implies that C is non-negligible, as required. ■

LEMMA 13.4.4. *For any finitely presented semigroup \mathfrak{S} the word problem in \mathfrak{S}_x is the C -clone of the word problem in \mathfrak{S} :*

$$WP_{\mathfrak{S}_x} = C(WP_{\mathfrak{S}}).$$

Proof. It follows immediately from Lemma 13.4.1. ■

THEOREM 13.4.5. *If the word problem in \mathfrak{S} is undecidable then the word problem in \mathfrak{S}_x is super-undecidable.*

Proof. By Lemma 13.4.3 C is an effective non-negligible cloning. By Lemma 13.4.4 $WP_{\mathfrak{S}_x} = C(WP_{\mathfrak{S}})$. Now by Theorem 13.3.3 if the word problem $WP_{\mathfrak{S}}$ is undecidable then the word problem $WP_{\mathfrak{S}_x} = C(WP_{\mathfrak{S}})$ is super-undecidable, as claimed. ■

EXAMPLE 13.4.6. In 1956 Tseitin constructed a semigroup \mathfrak{T} presented by 5 generators and 7 relations with unsolvable word problem [272]:

$$\mathfrak{T} = \langle a, b, c, d, e \mid ca, ad = da, bc = cb, bd = db, ce = eca, de = edb, cca = ccae \rangle.$$

In this case the super-undecidable semigroup \mathfrak{T}_x has 6 generators and 13 relators whose total length is equal to 49.

PROBLEM 13.4.7. Is there a finitely presented group with generically undecidable word problem?

13.5. Absolutely undecidable problems

In this section we discuss problems which are undecidable on every non-negligible subset. More precisely, let $I = \{0, 1\}^*$ be a set of words in an alphabet $\{0, 1\}$. Denote by $I_n = \{0, 1\}^n$ the sphere of radius n in I , which consists of all words of length n . We assume that I_n comes equipped with a uniform distribution ρ_n . The asymptotic density of a subset $S \subseteq I$ is the following limit (if it exists)

$$\rho(S) = \lim_{n \rightarrow \infty} \rho_n(S \cap I_n).$$

We say that the membership problem for S in I is *absolutely undecidable* (with respect to the density ρ) if there is no partial decision algorithm \mathcal{A} that correctly decides the membership problem for S and the halting set $H_{\mathcal{A}}$ of \mathcal{A} has positive asymptotic density $\rho(H_{\mathcal{A}}) > 0$. In this event we also say that S is absolutely undecidable. To strengthen our results we introduce the upper limit density:

$$\bar{\rho} = \limsup_{n \rightarrow \infty} \rho_n(S \cap I_n).$$

In this case every set S has well-defined asymptotic density $\bar{\rho}(S)$ and $\bar{\rho}(S) \geq \rho(S)$. It follows that if the membership problem for S is absolutely undecidable with respect to $\bar{\rho}$ then so it is relative to ρ . From now on we consider absolute decidability relative to $\bar{\rho}$. Notice, that S is absolutely undecidable if and only if it is undecidable on every non-negligible set. In this section we show that absolutely undecidable sets exist.

Recall that a subset $S \subseteq I$ is called *immune* if S is infinite and does not contain any infinite c.e. subset of I .

LEMMA 13.5.1. *There exists a generic (relative to ρ) immune subset of I .*

Proof. Let S_1, S_2, \dots be a sequence of all infinite c.e. subsets of I . We construct by induction a sequence of subsets A_1, A_2, \dots of I and a sequence of elements w_1, w_2, \dots of I . Let $A_0 = I$. Take an element $w_1 \in S_1$ and define $A_1 = A_0 - \{w_1\}$. Suppose by induction sets A_1, \dots, A_n and elements w_1, \dots, w_n are constructed such that for every $i = 1, \dots, n$ the following hold:

- 1) $w_i \in S_i$,
- 2) $A_i = A_{i-1} - \{w_i\}$,
- 3) $|w_{i-1}| < |w_i|$.

Since the set $\bigcup_{i=1}^n I_i$ is finite and the set S_{n+1} is infinite then there is an element $w_{n+1} \in S_{n+1}$ such that 3) holds. Define $A_{n+1} = A_n - \{w_{n+1}\}$. Then the conditions 1)-3) hold for $i = 1, \dots, n+1$. This finishes induction.

Put

$$A = \bigcap_{i=1}^{\infty} A_i.$$

By construction $S_n \not\subseteq A$ for every $n = 1, 2, \dots$. Clearly, A is infinite, since constructing A we removed at most one element from each sphere I_n . Moreover, A is generic (relative to ρ) since

$$\frac{|A \cap I_n|}{|I_n|} \geq \frac{2^n - 1}{2^n},$$

so $\rho(A) = 1$. ■

THEOREM 13.5.2. *Every generic immune subset of I is absolutely undecidable.*

Proof. Let A be a generic immune subset of I and \mathcal{A} a partial decision algorithm for the membership problem for A . If the halting set $H_{\mathcal{A}}$ of \mathcal{A} is non-negligible (with respect to $\bar{\rho}$) then the set $A \cap H_{\mathcal{A}}$ is also non-negligible. Indeed, since A is generic its complement \bar{A} is negligible, so $\limsup \rho_n(\bar{A} \cap I_n) = 0$. In this case

$$\rho_n(H_{\mathcal{A}} \cap I_n) = \rho_n(A \cap H_{\mathcal{A}} \cap I_n) + \rho_n(\bar{A} \cap H_{\mathcal{A}} \cap I_n).$$

Hence

$$\limsup \rho_n(A \cap H_{\mathcal{A}} \cap I_n) = \limsup \rho_n(H_{\mathcal{A}} \cap I_n) > 0,$$

as claimed. In particular, $A \cap H_{\mathcal{A}}$ is infinite. Notice, also that $A \cap H_{\mathcal{A}}$ is c.e., because $w \in A \cap H_{\mathcal{A}}$ if and only if \mathcal{A} halts on the input w and outputs 1. In this case, one can design a new algorithm \mathcal{A}' such that on an arbitrary input w \mathcal{A}' starts the algorithm \mathcal{A} and if it halts on w and gives the value 1 then \mathcal{A}' outputs this value, in all other cases \mathcal{A}' does not halt on w .

Thus $A \cap H_{\mathcal{A}}$ is an infinite c.e. subset of I which is contained in the immune set A — contradiction. \blacksquare

COROLLARY 13.5.3. *Absolutely undecidable subsets of I exist.*

Part 4

Asymptotically Dominant Properties and Cryptanalysis

Most modern cryptographic schemes use algebraic systems such as rings, groups, lattices, etc., as platforms. Typically, cryptographic protocols involve a random choice of various algebraic objects related to the platforms, like elements, or subgroups, or homomorphisms. One of the key points in using randomness is to foil various statistical attacks, or attacks which could use some specific properties of objects if the latter are not chosen randomly. The main goal of this chapter is to show that randomly chosen objects quite often have very special properties, which yields some “unexpected” attacks. We argue that the knowledge of basic properties of random objects must be part of any serious cryptanalysis and it has to be taken into consideration while choosing “good” keys.

In the paper [196] the authors study asymptotic properties of words representing the trivial element in a given finitely presented group G . It turns out that a randomly chosen trivial word in G has a “hyperbolic” van Kampen diagram, even if the group G itself is not hyperbolic. This allows one to design a correct (no errors) search/decision algorithm which solves the word (search) problem in polynomial time on a generic subset (i.e., on “most” elements) of a given group G . A similar result for the conjugacy search problem in finitely presented groups has been proven in [195] (we refer to [274] for a detailed analysis of the generic complexity of the word and conjugacy search problems in groups). These results show that group-based cryptographic schemes whose security is based on the word or conjugacy search problems are subject to effective attacks, unless the keys are chosen in the complements of the corresponding generic sets.

In this chapter we study asymptotic properties of finitely generated subgroups of groups. We start by introducing a methodology to deal with asymptotic properties of subgroups in a given finitely generated group, then we describe two such properties, and finally we show how one can use them in cryptanalysis of group based cryptographic schemes. Then we dwell on the role of asymptotically dominant properties of subgroups in modern cryptanalysis. We mostly focus on one particular example, the Anshel-Anshel-Goldfeld (AAG) key establishment protocol [5]; however, it seems plausible that similar analysis can be applied to some other cryptographic schemes as well. One of our main goals here is to give mathematical reasons behind surprisingly high success rates of the so-called *length based attacks* in breaking the AAG protocol. Another goal is to analyze an attack that we call a *quotient attack* (cf. our Chapter 15). We also want to emphasize that we believe that this “asymptotic cryptanalysis” suggests a way to choose strong keys (like groups, subgroups, or elements) for the general AAG scheme (with different groups as the platforms) that may foil some of the known attacks, including the ones discussed here.

Our main focus in this part of the book is on the length based attacks (LBA). This idea appeared first in the paper [136] by Hughes and Tannenbaum, and later was further developed in two papers [86] and [87] by Garber, Kaplan, Teicher, Tsaban, and Vishne. Recently, the most successful version of this attack for the (simultaneous) conjugacy search problem in braid groups was developed in [211]. We note that Ruinsky, Shamir, and Tsaban used LBA in attacking some other algorithmic problems in groups [240].

The basic idea of LBA is very simple. One solves the simultaneous conjugacy search problem relative to a subgroup (SCSP*) (i.e., with a constraint that solutions should be in a given subgroup) precisely the same way as this would be done in a

free group. Surprisingly, experiments show that this strategy works well in groups which seem to be far from being free, for instance, in braid groups. We claim that the primary reason behind this phenomenon is that asymptotically, finitely generated subgroups in many groups are free. More precisely, in many groups a randomly chosen finite set of elements freely generates a free subgroup with overwhelming probability. We say that such groups have the *free basis property*. This allows one to analyze the generic-case complexity of LBA, SCSP*, and some other related algorithmic problems. Moreover, we argue that LBA implicitly relies on fast computing of the geodesic length of elements in finitely generated subgroups of the platform group, or of some good approximations of that length. In fact, most LBA strategies tacitly assume that the geodesic length of elements in a given group is a good approximation of the geodesic length of the same elements in a subgroup. At first it may look like a wrong assumption, because it is known that even in a braid group B_n , $n \geq 3$, there are infinitely many subgroups whose distortion function (which measures the geodesic length of elements in a subgroup relative to that in G) is not bounded by any recursive function. We show, nevertheless, that in many groups the distortion of randomly chosen finitely generated subgroups is linear. Our main focus is on the braid group B_n , $n \geq 3$. We establish our main results here for the pure braid group PB_n , which is a subgroup of finite index in the ambient braid group B_n . We conjecture that the results hold in the group B_n as well, and hope to fill in this gap in the near future. We also mention that our results hold for all finitely generated groups that have non-abelian free quotients.

While studying length based attacks, we realized that quotient attacks (QA) appear to be yet another class of powerful attacks on the AAG scheme. These attacks are just fast generic algorithms to solve various search problems in groups; cf. our Chapter 15. The main idea behind QA is the following observation: to solve a computational problem in a group G it suffices, on most inputs, to solve it in a suitable quotient G/N , provided G/N has a fast decision algorithm for the problem. Robustness of such an algorithm relies on the following property of the quotient G/N : a randomly chosen finitely generated subgroup of G has trivial intersection with the kernel N . In particular, this is the case if G/N is a free non-abelian group. We note that a similar idea was already exploited in [146], but there the answer was given only for inputs in the “No” part of a given decision problem, which does not apply to search problems at all. The strength of our approach comes from the extra requirement that G/N has the free basis property.

To conclude these introductory remarks, we say that we believe that the AAG scheme, despite being heavily battered by several attacks, is still very much “alive”. It simply did not get a fair chance to survive because of the insufficient group-theoretic research it required. It is still quite plausible that there are platform groups and methods to choose strong keys for AAG that would foil all known attacks. To find such a platform group is an interesting and challenging algebraic problem. We emphasize once again that our method of analyzing generic-case complexity of computational security assumptions of the AAG scheme, based on the asymptotic behavior of subgroups in a given group, creates a bridge between asymptotic algebra and cryptanalysis. Also, this method can be applied to some other cryptographic schemes as well.

CHAPTER 14

Asymptotically Dominant Properties

In this chapter we develop some tools to study asymptotic properties of subgroups of groups. Throughout this chapter by G we denote a group with a finite generating set X .

14.1. A brief description

Asymptotic properties of subgroups, a priori, depend on a given probability distribution on these subgroups. In general, there are several natural procedures to generate subgroups in a given group. However, there is no unique universal distribution of this kind. We refer to [97] for a discussion on different approaches to random subgroup generation.

Our basic principle here is that in applications, one has to consider a particular distribution that comes from a particular problem addressed in the given application, say in a cryptographic protocol. As soon as the distribution is fixed, one can approach asymptotic properties of subgroups via asymptotic densities with respect to a fixed stratification of the set of subgroups. We briefly discuss these ideas below and refer to [13, 29, 146, 147], and to a recent survey [98], for a thorough exposition. In Section 14.2, we adjust these general ideas to a particular way to generate subgroups that is used in cryptography.

The first step is to choose and fix a particular way to describe finitely generated subgroups H of the group G . For example, a description δ of H could be a tuple of words (u_1, \dots, u_k) in the alphabet $X^{\pm 1} = X \cup X^{-1}$ representing a set of generators of H , or a folded finite graph that accepts the subgroup generated by the generators $\{u_1, \dots, u_k\}$ of H in the ambient free group $F(X)$ (see [145]), etc. In general, the descriptions above are by no means unique for a given subgroup H .

When the way of describing subgroups in G is fixed, one can consider the set Δ of all such descriptions of all finitely generated subgroups of G . The next step is to define a *size* $s(\delta)$ of a given description $\delta \in \Delta$, i.e., a function

$$s : \Delta \rightarrow \mathbb{N}$$

in such a way that the set (the ball of radius n)

$$B_n = \{\delta \in \Delta \mid s(\delta) \leq n\}$$

is finite. This gives a *stratification* of the set Δ into a union of finite balls:

$$(19) \quad \Delta = \bigcup_{n=1}^{\infty} B_n.$$

Let μ_n be a given probabilistic measure on B_n (it could be the measure induced on B_n by some fixed measure on the whole set Δ or a measure not related to any

measure on Δ). The stratification (36) and the collection of measures

$$(20) \quad \{\mu_n\} = \{\mu_n \mid n \in \mathbb{N}\}$$

allow one to estimate the asymptotic behavior of subsets of Δ . For a subset $R \subseteq \Delta$, the *asymptotic density* $\rho_\mu(R)$ is defined by the following limit (if it exists)

$$\rho_\mu(R) = \lim_{n \rightarrow \infty} \mu_n(R \cap B_n).$$

If μ_n is the uniform distribution on the finite set B_n then

$$\mu_n(R \cap B_n) = \frac{|R \cap B_n|}{|B_n|}$$

is the n -th *frequency*, or probability, to hit an element from R in the ball B_n . In this case we refer to $\rho_\mu(R)$ as to the *asymptotic density* of R and denote it by $\rho(R)$.

One can also define the asymptotic density using \limsup rather than \lim , in which case $\rho_\mu(R)$ always exists.

We say that a subset $R \subseteq \Delta$ is *generic* if $\rho_\mu(R) = 1$ and *negligible* if $\rho_\mu(R) = 0$. It is worthwhile to mention that the asymptotic densities not only allow one to distinguish between “large” (generic) and “small” (negligible) sets, but also give a tool to distinguish various large (or small) sets. For instance, we say that R has asymptotic density $\rho_\mu(R)$ with a *super-polynomial convergence rate* if

$$|\rho_\mu(R) - \mu_n(R \cap B_n)| = o(n^{-k})$$

for any $k \in \mathbb{N}$. For brevity, R is called *strongly generic* if $\rho_\mu(R) = 1$ with a super-polynomial convergence rate. The set R is *strongly negligible* if its complement $S - R$ is strongly generic.

Similarly, one can define exponential convergence rates and exponentially generic (negligible) sets.

14.2. Random subgroups and generating tuples

In this section we review a procedure, which is most commonly used in group-based cryptography to generate random subgroups of a given group (see e.g. [5]). Briefly, the procedure is as follows.

Random generation of subgroups in G :

- pick a random $k \in \mathbb{N}$ within given bounds: $K_0 \leq k \leq K_1$;
- pick random k words $w_1, \dots, w_k \in F(X)$ with fixed length range: $L_0 \leq |w_i| \leq L_1$;
- output the subgroup $\langle w_1, \dots, w_k \rangle$ of G .

Without loss of generality we may fix, from the beginning, a single natural number k , instead of choosing it from a finite interval $[K_0, K_1]$ (by the formula of complete probability the general case can be reduced to this one). Fix $k \in \mathbb{N}$, $k \geq 1$, and focus on the set of all k -generated subgroups of G .

The corresponding descriptions δ , the size function, and the corresponding stratification of the set of all descriptions can be formalized as follows. By a description $\delta(H)$ of a k -generated subgroup H of G we understand here any k -tuple (w_1, \dots, w_k) of words from $F(X)$ that generates H in G . Hence, in this case the space of all descriptions is the Cartesian product $F(X)^k$ of k copies of $F(X)$:

$$\Delta = \Delta_k = F(X)^k.$$

The size $s(w_1, \dots, w_k)$ can be defined either as the total length of the generators

$$s(w_1, \dots, w_k) = |w_1| + \dots + |w_k|,$$

or as the maximum length of the components:

$$s(w_1, \dots, w_k) = \max\{|w_1|, \dots, |w_k|\}.$$

Our approach works for both definitions, so we do not specify which one we use here. For $n \in \mathbb{N}$, denote by B_n the ball of radius n in Δ :

$$B_n = \{(w_1, \dots, w_k) \in F(X)^k \mid s(w_1, \dots, w_k) \leq n\}.$$

This gives the required stratification

$$\Delta = \bigcup_{n=1}^{\infty} B_n.$$

For a subset M of Δ we define the asymptotic density $\rho(M)$ relative to the stratification above assuming the uniform distribution on the balls B_n :

$$\rho(M) = \lim_{n \rightarrow \infty} \frac{|B_n \cap M|}{|B_n|}.$$

We note that there are several obvious deficiencies in this approach: (1) we consider subgroups with a fixed number of generators; (2) a subgroup typically has many different k -generating tuples; (3) every generator can be described by several different words from $F(X)$. Thus, our descriptions are far from being unique. However, as we have mentioned before, these models reflect standard methods to generate subgroups in cryptographic protocols. We refer to [97] for other approaches.

14.3. Asymptotic properties of subgroups

Let G be a group with a finite set of generators X , and k a fixed positive integer. Denote by \mathcal{P} a property of descriptions of k -generated subgroups of G . By $\mathcal{P}(G)$ we denote the set of all descriptions from $\Delta = \Delta_k$ that satisfy \mathcal{P} in G .

DEFINITION 14.3.1. We say that a property $\mathcal{P} \subseteq \Delta$ of descriptions of k -generated subgroups of G is:

- 1) *asymptotically visible* in G if $\rho(\mathcal{P}(G)) > 0$;
- 2) *generic* in G if $\rho(\mathcal{P}(G)) = 1$;
- 3) *strongly generic* in G if $\rho(\mathcal{P}(G)) = 1$, and the rate of convergence of $\rho_n(\mathcal{P}(G))$ is super-polynomial;
- 4) *exponentially generic* in G if $\rho(\mathcal{P}(G)) = 1$ and the rate of convergence of $\rho_n(\mathcal{P}(G))$ is exponential.

Informally, if \mathcal{P} is asymptotically visible for k -generated subgroups of G , then there is a certain non-zero probability that a randomly and uniformly chosen description $\delta \in \Delta$ of a sufficiently big size results in a subgroup of G satisfying \mathcal{P} . Similarly, if \mathcal{P} is exponentially generic for k -generated subgroups of G , then a randomly and uniformly chosen description $\delta \in \Delta$ of a sufficiently big size results in a subgroup of G satisfying \mathcal{P} with overwhelming probability. Likewise, one can interpret generic and strongly generic properties of subgroups. If a set of descriptions Δ of subgroups of G is fixed, then we sometimes abuse the terminology and refer to asymptotic properties of descriptions of subgroups as asymptotic properties of the subgroups itself.

EXAMPLE 14.3.2. Let H be a fixed k -generated group. Consider the following property \mathcal{P}_H : a given description $(w_1, \dots, w_k) \in F(X)^k$ satisfies \mathcal{P}_H if the subgroup $\langle w_1, \dots, w_k \rangle$ generated in G by this tuple, is isomorphic to H . If $\mathcal{P}_H(G)$ is asymptotically visible (generic) in Δ , then we say that the group H is asymptotically visible (generic) in G (among k -generated subgroups).

By k -spectrum $Spec_k(G)$ of G we denote the set of all (up to isomorphism) k -generated subgroups that are asymptotically visible in G .

There are several natural problems about asymptotically visible subgroups of G that play an important role in cryptography. For example, when choosing k -generated subgroups of G randomly, it might be useful to know what kind of subgroups you can get with non-zero probability. Hence the following problem is of interest:

PROBLEM 14.3.3. What is the spectrum $Spec_k(G)$ for a given group G and a natural number $k \geq 1$?

More technical, but also important in applications is the following problem.

PROBLEM 14.3.4. Does the spectrum $Spec_k(G)$ depend on a given finite set of generators of G ?

We will see in due course that knowing the answers to these problems is important for choosing strong keys in some group-based cryptographic schemes.

14.4. Groups with generic free basis property

DEFINITION 14.4.1. We say that a tuple $(u_1, \dots, u_k) \in F(X)^k$ has the *free basis* property (\mathcal{FB}) in G if it freely generates a free subgroup in G .

In [142] Jitsukawa showed that \mathcal{FB} is generic for k -generated subgroups of a finitely generated free non-abelian group $F(X)$ for every $k \geq 1$ with respect to the standard basis X . Martino, Turner and Ventura strengthened this result in [185] by proving that \mathcal{FB} is *exponentially generic* in $F(X)$ for every $k \geq 1$ with respect to the standard basis X . Recently, it has been shown in [97] that \mathcal{FB} is exponentially generic in arbitrary hyperbolic non-elementary (in particular, free non-abelian) group for every $k \geq 1$ and with respect to any finite set of generators.

We say that a group G has the *generic free basis* property if \mathcal{FB} is generic in G for every $k \geq 1$ and every finite generating set of G . In a similar way, we define groups with *strongly* and *exponentially* generic free basis property. By \mathcal{FB}_{gen} , \mathcal{FB}_{st} , \mathcal{FB}_{exp} , we denote classes of finitely generated groups with, respectively, generic, strongly generic, and exponentially generic free basis property.

The following result gives a host of examples of groups with generic \mathcal{FB} .

THEOREM 14.4.2. *Let G be a finitely generated group, and N a normal subgroup of G . If the quotient group G/N is in the class \mathcal{FB}_{gen} , or in \mathcal{FB}_{st} , or in \mathcal{FB}_{exp} , then the whole group G is in the same class.*

Proof. Let $H = G/N$, and let $\varphi : G \rightarrow H$ be the canonical epimorphism. Fix a finite generating set X of G and a natural number $k \geq 1$. Clearly, X^φ is a finite generating set of H . By our assumption, the free basis property is generic in H with respect to the generating set X^φ and given k . By identifying $x \in X$ with $x^\varphi \in H$ we may assume that a finitely generated subgroup A of G and the subgroup A^φ have the same set of descriptions. Observe now that a subgroup A of G generated

by a k -tuple $(u_1, \dots, u_k) \in F(X)^k$ has the following property: if A^φ is free with basis $(u_1^\varphi, \dots, u_k^\varphi)$, then A is free with basis (u_1, \dots, u_k) . Therefore, for each $t \in \mathbb{N}$ one has

$$\frac{|B_t \cap \mathcal{FB}(G)|}{|B_t|} \geq \frac{|B_t \cap \mathcal{FB}(H)|}{|B_t|}.$$

This implies that if $\mathcal{FB}(H)$ is generic in $H = G/N$, then $\mathcal{FB}(G)$ is generic in G , and its convergence rate in G is not less than the corresponding convergence rate in H , as claimed. ■

The result above bears on some infinite groups recently used in group-based cryptography. Braid groups B_n appear as one of the main platforms in braid-group cryptography so far (see [5, 161, 58, 6]). Recall (see our Section 5.1) that the braid group B_n can be defined by the classical Artin presentation:

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{ll} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j & \text{if } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i & \text{if } |i - j| > 1 \end{array} \right\rangle.$$

Denote by $\sigma_{i,i+1}$ the transposition $(i, i + 1)$ in the symmetric group Σ_n . The map $\sigma_i \rightarrow \sigma_{i,i+1}$, $i = 1, \dots, n$ gives rise to the canonical epimorphism $\pi : B_n \rightarrow \Sigma_n$. The kernel of π is a subgroup of index $n!$ in B_n , called the *pure braid group* PB_n .

COROLLARY 14.4.3. *The free basis property is exponentially generic in the pure braid group PB_n for any $n \geq 3$.*

Proof. It is known (see e.g. [21]) that the group PB_n , $n \geq 3$, maps onto PB_3 , and the group PB_3 is isomorphic to $F_2 \times \mathbb{Z}$. Therefore, PB_n , $n \geq 3$, maps onto the free group F_2 . Now the result follows from Theorem 14.4.2 and the strong version of Jitsukawa's result mentioned above [185, 97, 142]. ■

Although we have shown that the pure braid group PB_n , $n \geq 3$, has exponentially generic free basis property and is a subgroup of finite index in the braid group B_n , at the moment we cannot prove that B_n has exponentially generic free basis property, so we ask:

PROBLEM 14.4.4. Is it true that the braid groups B_n , $n \geq 3$, have exponentially generic free basis property?

In [180], partially commutative groups were proposed as possible platforms for some cryptographic schemes. We refer to [20] for a more recent discussion on this. By definition, a partially commutative group $G(\Gamma)$ (also known as right angled Artin group, cf. our Section 5.6) is a group associated with a finite graph $\Gamma = (V, E)$, with a set of vertices $V = \{v_1, \dots, v_n\}$ and a set of edges $E \subseteq V \times V$, given by the following presentation:

$$G(\Gamma) = \langle v_1, \dots, v_n \mid v_i v_j = v_j v_i \text{ for } (v_i, v_j) \in E \rangle.$$

Note that the group $G(\Gamma)$ is abelian if and only if the graph Γ is complete.

COROLLARY 14.4.5. *The free basis property is exponentially generic in non-abelian partially commutative groups.*

Proof. Let $G = G(\Gamma)$ be a non-abelian partially commutative group corresponding to a finite graph Γ . Then there are three vertices in Γ , say v_1, v_2, v_3 such that the subgraph Γ_0 of Γ on these vertices is not a triangle. In particular, the partially commutative group $G_0 = G(\Gamma_0)$ is either a free group F_3 (no edges in Γ_0), or $(\mathbb{Z} \times \mathbb{Z}) * \mathbb{Z}$ (only one edge in Γ_0), or $F_2 \times \mathbb{Z}$ (precisely two edges in Γ_0). In all

three cases the group $G(\Gamma_0)$ maps onto F_2 . Now it suffices to observe that $G(\Gamma)$ maps onto $G(\Gamma_0)$ since $G(\Gamma_0)$ is obtained from $G(\Gamma)$ by adding to the standard presentation of $G(\Gamma)$ all the relations of type $v = 1$, where v is a vertex of Γ different from v_1, v_2, v_3 . This shows that F_2 is a quotient of $G(\Gamma)$ and the result follows from Theorem 14.4.2. ■

Note that some other groups that have been proposed as platforms in group-based cryptography, do not have non-abelian free subgroups at all, so they do not have free basis property for $k \geq 2$. For instance, in [223] Grigorchuk's groups were used as platforms. Since these groups are periodic (i.e., every element has finite order), they do not contain free subgroups. It is not clear what the asymptotically visible subgroups in Grigorchuk groups are. As another example, note that in [250], the authors propose Thompson's group F as a platform. It is known that there are no non-abelian free subgroups in F (see, for example, [34]), so F does not have free basis property. Recently, some interesting results on the spectrum $Spec_k(F)$ were obtained in [40].

14.5. Quasi-isometrically embedded subgroups

In this section we discuss another property of subgroups of G that plays an important role in our cryptanalysis of group-based cryptographic schemes.

Let G be a group with a finite generating set X . The *Cayley graph* $\Gamma(G, X)$ is an X -labeled directed graph with the vertex set G and such that any two vertices $g, h \in G$ are connected by an edge from g to h with a label $x \in X$ if and only if $gx = h$ in G . For convenience we usually assume that the set X is closed under inversion, i.e., $x^{-1} \in X$ for every $x \in X$. One can introduce a metric d_X on G setting $d_X(g, h)$ equal to the length of a shortest word in $X = X \cup X^{-1}$ representing the element $g^{-1}h$ in G . It is easy to see that $d_X(g, h)$ is equal to the length of a shortest path from g to h in the Cayley graph $\Gamma(G, X)$. This turns G into a metric space (G, d_X) . By $l_X(g)$ we denote the length of a shortest word in generators X representing the element g . Clearly, $l_X(g) = d_X(1, g)$.

Let H be a subgroup of G generated by a finite set Y of elements. Then there are two metrics on H : one is d_Y described above and the other one is the metric d_X induced from the metric space (G, d_X) on the subspace H . The following notion allows one to compare these metrics. Recall that a map $f : M_1 \rightarrow M_2$ between two metric spaces (M_1, d_1) and (M_2, d_2) is a *quasi-isometric embedding* if there are constants $\lambda > 1, c > 0$ such that for every elements $x, y \in M_1$ the following inequalities hold:

$$(21) \quad \frac{1}{\lambda}d_1(x, y) - c \leq d_2(f(x), f(y)) \leq \lambda d_1(x, y) + c.$$

In particular, we say that a subgroup H with a finite set of generators Y is *quasi-isometrically embedded* into G if the inclusion map $i : H \hookrightarrow G$ is a quasi-isometric embedding $i : (H, d_Y) \rightarrow (G, d_X)$. Note that in this case the right-hand inequality in (21) always holds, since for all $f, h \in H$ one has

$$d_X(i(f), i(h)) \leq \max_{y \in Y} \{l_X(y)\} \cdot d_Y(f, h).$$

Therefore, the definition of a quasi-isometrically embedded subgroup takes the following simple form (in the notation above).

DEFINITION 14.5.1. Let G be a group with a finite generating set X , and H a subgroup of G generated by a finite set of elements Y . Then H is *quasi-isometrically embedded* in G if there are constants $\lambda > 1, c > 0$ such that for all elements $f, h \in H$ the following inequality holds:

$$(22) \quad \frac{1}{\lambda}d_Y(f, h) - c \leq d_X(f, h).$$

It follows immediately from the definition that if X and X' are two finite generating sets of G , then the metric spaces (G, d_X) and $(G, d_{X'})$ are quasi-isometrically embedded into each other. This implies that the notion of quasi-isometrically embedded subgroups is independent of the choice of finite generating sets in H or in G (though the constants λ and c could be different).

DEFINITION 14.5.2. Let G be a group with a finite generating set X . We say that a tuple $(u_1, \dots, u_k) \in F(X)^k$ has a \mathcal{QI} (quasi-isometric embedding) property in G if the subgroup it generates in G is quasi-isometrically embedded in G .

Denote by $\mathcal{QI}(G)$ the set of all tuples in $F(X)^k$ that satisfy the \mathcal{QI} property in G . We say that the property \mathcal{QI} is *generic* in G if $\mathcal{QI}(G)$ is generic in G for every $k \geq 1$ and every finite generating set of G . Similarly, we define groups with *strongly* and *exponentially* generic quasi-isometric embedding subgroup property. Denote by $\mathcal{QI}_{gen}, \mathcal{QI}_{st}, \mathcal{QI}_{exp}$ classes of finitely generated groups with, respectively, generic, strongly generic, and exponentially generic quasi-isometric embedding subgroup property.

It is not hard to see that *every* finitely generated subgroup of a finitely generated free group F is quasi-isometrically embedded in F , so $F \in \mathcal{QI}_{exp}$.

The following result gives further examples of groups with quasi-isometric embedding subgroup property.

Let $G \in \mathcal{FB}_{gen} \cap \mathcal{QI}_{gen}$. Note that the intersection of two generic sets $\mathcal{FB}(G) \subseteq F(X)^k$ and $\mathcal{QI}(G) \subseteq F(X)^k$ is again a generic set in $F(X)^k$, so the set $\mathcal{FB}(G) \cap \mathcal{QI}(G)$ of all tuples $(u_1, \dots, u_k) \in F(X)^k$ that freely generate a quasi-isometrically embedded subgroup of G , is generic in $F(X)^k$. Observe that by the remark above and by the result on free basis property in free groups, $\mathcal{FB}_{gen} \cap \mathcal{QI}_{gen}$ contains all free groups of finite rank. The argument applies also to the strongly generic and exponentially generic versions of the properties. To unify references we will use the following notation: $\mathcal{FB}_* \cap \mathcal{QI}_*$, for $* \in \{gen, st, exp\}$.

THEOREM 14.5.3. *Let G be a finitely generated group with a quotient G/N . If $G/N \in \mathcal{FB}_* \cap \mathcal{QI}_*$ then $G \in \mathcal{FB}_* \cap \mathcal{QI}_*$ for any $* \in \{gen, st, exp\}$.*

Proof. Let G be a finitely generated group generated by X , N a normal subgroup of G such that the quotient G/N is in $\mathcal{FB}_* \cap \mathcal{QI}_*$. Let $\varphi : G \rightarrow G/N$ be the canonical epimorphism. By Theorem 14.4.2, $G \in \mathcal{FB}_*$, so it suffices to show now that $G \in \mathcal{QI}_*$.

Let H be a k -generated subgroup with a set of generators $Y = (u_1, \dots, u_k) \in F(X)^k$. Suppose that $Y \in \mathcal{FB}_*(G/N) \cap \mathcal{QI}_*(G/N)$, i.e., the image Y^φ of Y in G/N freely generates a free group quasi-isometrically embedded into G/N . Observe first that for every element $g \in G$, one has $l_X(g) \geq l_{X^\varphi}(g^\varphi)$, where l_{X^φ} is the length function on G/N relative to the set X^φ of generators. Since the subgroup H^φ is quasi-isometrically embedded into G/N , the metric space $(H^\varphi, d_{Y^\varphi})$ quasi-isometrically embeds into $(G^\varphi, d_{X^\varphi})$. On the other hand, φ maps the subgroup H

onto the subgroup H^φ isomorphically (since both are free groups with the corresponding bases), such that for any $h \in H$, one has $d_Y(h) = d_{Y^\varphi}(h^\varphi)$. Now we can deduce the following inequalities for $g, h \in H$:

$$\frac{1}{\lambda}d_Y(g, h) - c = \frac{1}{\lambda}d_{Y^\varphi}(g^\varphi, h^\varphi) - c \leq d_{X^\varphi}(g^\varphi, h^\varphi) \leq d_X(g, h),$$

where λ and c come from the quasi-isometric embedding of H^φ into G/N . This shows that H is quasi-isometrically embedded in G , as required. ■

COROLLARY 14.5.4. *The following groups are in $\mathcal{FB}_{exp} \cap \mathcal{QI}_{exp}$:*

- 1) *Pure braid groups PB_n , $n \geq 3$;*
- 2) *Non-abelian partially commutative groups $G(\Gamma)$.*

Proof. The arguments in Corollaries 14.4.3, 14.4.5 show that the groups PB_n , $n \geq 3$, and $G(\Gamma)$ are non-commutative and have quotient isomorphic to the free group F_2 . Now the result follows from Theorems 14.4.2 and 14.5.3. ■

CHAPTER 15

Length Based and Quotient Attacks

15.1. Anshel-Anshel-Goldfeld scheme

In this section we discuss the Anshel-Anshel-Goldfeld (AAG) public key exchange protocol [5] and touch briefly on its security.

15.1.1. Description of the Anshel-Anshel-Goldfeld scheme. We have already described the AAG protocol in our Section 4.5, but we briefly describe it here again for the reader's convenience.

Let G be a group with a finite generating set X . G is called the *platform* of the scheme. We assume that elements w in G have unique normal forms \bar{w} such that it is hard to recover w from \bar{w} , and there is a fast algorithm to compute \bar{w} for any given w .

The Anshel-Anshel-Goldfeld key exchange protocol is the following sequence of steps. Alice [Bob, respectively] chooses a random subgroup of G ,

$$A = \langle a_1, \dots, a_m \rangle \quad [B = \langle b_1, \dots, b_n \rangle \text{ respectively}]$$

by randomly choosing generators a_1, \dots, a_m [b_1, \dots, b_n respectively] as words in $X^{\pm 1}$, and makes it public. Then Alice [Bob, respectively] chooses randomly a secret element $a = u(a_1, \dots, a_m) \in A$ [$b = v(b_1, \dots, b_n) \in B$ respectively] as a product of the generators of A [B respectively] and their inverses, takes the conjugates b_i^a, \dots, b_n^a [a_1^b, \dots, a_m^b respectively], diffuses them by taking their normal forms \bar{b}_i^a [\bar{a}_j^b respectively], and makes these normal forms public:

$$\bar{b}_1^a, \dots, \bar{b}_n^a \quad [\bar{a}_1^b, \dots, \bar{a}_m^b \text{ respectively}].$$

Afterwards, Alice [Bob, respectively] computes $a^{-1}a^b$ [$(b^a)^{-1}b$ respectively] and takes its normal form. Since

$$a^{-1}a^b = [a, b] = (b^a)^{-1}b,$$

the obtained normal form is the shared secret key.

15.1.2. Security assumptions of the AAG scheme. In this section we briefly discuss computational security features of the AAG scheme. Unfortunately, in the original description, the authors of the scheme did not state precisely the security assumptions that should make the scheme difficult to break. Here we dwell on several possible assumptions of this type, that often occur, though sometimes implicitly, in the literature on the AAG scheme.

It appears that the security of the AAG scheme relies on the computational hardness of the following computational problem in group theory:

AAG Problem: Given all of the public information from the AAG scheme, i.e., the group G , the elements $a_1, \dots, a_m, b_1, \dots, b_n$, and $\bar{b}_1^a, \dots, \bar{b}_n^a, \bar{a}_1^b, \dots, \bar{a}_n^b$ in the group G , find the shared secret key $[a, b]$.

This problem is not a standard group-theoretic problem, so not much is known about its complexity, and it is quite technical to formulate. So it would be convenient to reduce this problem to some standard algorithmic problem about groups or to a combination of such problems. The following problems seem to be relevant here and they have attracted quite a lot of attention recently, especially in the braid groups context (braid groups were the original platform for AAG [5]. We refer to the papers [33], [22], [24], [99], [168], [170]. Nevertheless, the precise relationship between these problems and AAG is unclear; see [252] for more details.

The Conjugacy Search Problem (CSP): Given $u, v \in G$ such that the equation $u^x = v$ has a solution in G , find a solution.

The Simultaneous Conjugacy Search Problem (SCSP): Given $u_i, v_i \in G$, such that the system $u_i^x = v_i$, $i = 1, \dots, n$ has a solution in G , find a solution.

The Simultaneous Conjugacy Search Problem relative to a subgroup (SCSP *): given $u_i, v_i \in G$ and a finitely generated subgroup A of G such that the system $u_i^x = v_i$, $i = 1, \dots, n$ has a solution in A , find a solution.

REMARK 15.1.1. Observe that if the word problem is decidable in G , then all of the problems above are also decidable. Indeed, one can enumerate all possible elements x (either in G or in the subgroup A), plug them one at a time into the equations and check, using the decision algorithm for the word problem in G , whether or not x is a solution. Since the systems above do have solutions, this algorithm will eventually find one. However, the main problem here is not decidability itself, the problem is whether or not one can find a solution sufficiently efficiently, say in polynomial time in the size of the inputs.

The following result is easy.

LEMMA 15.1.2. *For any group G , the AAG problem can be reduced in linear time to the SCSP * .*

Proof. Suppose in a finitely generated group G we are given the public data from the AAG scheme, i.e., we are given the subgroups

$$A = \langle a_1, \dots, a_m \rangle, \quad B = \langle b_1, \dots, b_n \rangle,$$

and the elements $\bar{b}_1^a, \dots, \bar{b}_n^a$ and $\bar{a}_1^b, \dots, \bar{a}_n^b$. If the SCSP relative to subgroups A and B is decidable in G , then by solving the system of equations

$$(23) \quad b_1^x = \bar{b}_1^a, \dots, b_n^x = \bar{b}_n^a$$

in A , one can find a solution $u \in A$. Similarly, by solving the system of equations

$$(24) \quad a_1^y = \bar{a}_1^b, \dots, a_m^y = \bar{a}_m^b$$

in B , one can find a solution $v \in B$. Note that all solutions of the system (23) are elements of the form ca , where c is an arbitrary element from the centralizer $C_G(B)$, and all solutions of the system (24) are of the form db for some $d \in C_G(A)$. In this case, obviously $[u, v] = [ca, db] = [a, b]$ gives a solution to the AAG problem. ■

Clearly, in some groups, for example, in abelian groups, the AAG problem as well as the SCSP* are both decidable in polynomial time, which makes them (formally) polynomial time equivalent. We will see in Section 15.2.2 that SCSP* is easy in free groups, too.

It is not clear, in general, whether the SCSP is any harder or easier than the CSP. In hyperbolic groups SCSP, as well as CSP, is easy [33]. There are indications that in finite simple groups, at least generically, the SCSP* is not harder than the standard CSP (since in those groups two randomly chosen elements generate the whole group). We refer to the preprint [98] for a brief discussion on complexity of these problems.

It would be interesting to make progress on the following problems, which would shed some light on the complexity of the AAG problem.

PROBLEM 15.1.3.

- 1) In which groups the AAG problem is poly-time equivalent to the SCSP*?
- 2) In which groups is the SCSP* harder than the SCSP?
- 3) In which groups is the SCSP harder (easier) than the CSP?

In the rest of this chapter we study the hardness of the SCSP* in various groups and analyze some of the most successful attacks on the AAG scheme from the viewpoint of asymptotic mathematics.

15.2. Length based attacks

The intuitive idea of the length based attack (LBA) was first put forward in the paper [136] by Hughes and Tannenbaum. In their paper [87], Garber, Kaplan, Teicher, Tsaban, and Vishne gave experimental results suggesting that very large computational power is required for this method to successfully solve the simultaneous conjugacy search problem. In [86], the same authors proposed a variant of the length based attack that uses memory, and gave experimental results showing that natural types of equations or system of equations in random subgroups of the braid groups can be solved, with high success rates, using the memory-length approach. However, the memory-length attacks were not tried in [86] against the actual parameters used in the AAG protocol. Recently, another variation of the length-based attack for braid groups was developed in [211], which turned out to be very successful against the AAG protocol. The authors of [211] suggested to use a heuristic algorithm for approximation of the geodesic length of braids in conjunction with LBA. Furthermore, they analyzed the reasons for success/failure of their variation of the attack, in particular, the practical importance of Alice's and Bob's subgroups A and B being non-isometrically embedded and being able to choose the elements of these subgroups distorted in the group (they refer to such elements as “peaks”).

In this section we rigorously prove that the same results can be observed in much larger classes of groups. In particular, our analysis works for the class \mathcal{FB}_{exp} , and hence for free groups, pure braid groups, locally commutative non-abelian groups, etc.

15.2.1. A general description. Since LBA is an attack on the AAG scheme, the inputs for LBA are precisely the inputs for the AAG algorithmic problem; see Section 15.1.2. Moreover, in all of its variations LBA attacks the AAG scheme by solving the corresponding conjugacy equations given in a particular instance of the

AAG problem. In what follows we take a slightly more general approach and view LBA as a correct partial deterministic search algorithm of a particular type for the simultaneous conjugacy search problem relative to a subgroup (SCSP*) in a given group G . In this case LBA is employed to solve the SCSP*, not the AAG problem. Below we describe a basic LBA in its most simplistic form.

Let G be a group with a finite generating set X . Suppose we are given a particular instance of the SCSP*, i.e., a system of conjugacy equations $u_i^x = v_i, i = 1, \dots, m$, which has a solution in a subgroup $A = \langle Y \rangle$ generated by a finite set Y of elements in G (given by words in $F(X)$). The task is to find such a solution in A . The main idea of LBA is very simple and it is based on the following assumptions:

(L1) for arbitrary “randomly chosen” elements $u, w \in G$, one has $l_X(u^w) > l_X(u)$,

which is convenient sometimes to have in a more general form:

(L2) for “randomly chosen” elements w, y_1, \dots, y_k in G , the element w has minimal l_X -length among all elements of the form w^y , where y runs over the subgroup of G generated by y_1, \dots, y_k .

It is not at all obvious whether these assumptions are actually correct for a given platform group. We will return to these issues in due course; now we just say that at least there has to be an algorithm \mathcal{A} to compute the length $l_X(w)$ for any given element $w \in G$.

Consider Alice’s public conjugates $\bar{b}_1^a, \dots, \bar{b}_n^a$, where $a = a_{s_1}^{\varepsilon_1} \dots a_{s_L}^{\varepsilon_L}$. Each \bar{b}_i^a is the result of a sequence of conjugations of b_i by generators of A :

$$(25) \quad \begin{array}{ccccc} & b_i & & & \\ & \downarrow & & & \\ a_{s_1}^{-\varepsilon_1} & b_i & a_{s_1}^{\varepsilon_1} & & \\ & \downarrow & & & \\ a_{s_2}^{-\varepsilon_2} a_{s_1}^{-\varepsilon_1} & b_i & a_{s_1}^{\varepsilon_1} a_{s_2}^{\varepsilon_2} & & \\ & \downarrow & & & \\ & \dots & & & \\ & \downarrow & & & \\ \bar{b}_i^a = & a_{s_L}^{-\varepsilon_L} \dots a_{s_2}^{-\varepsilon_2} a_{s_1}^{-\varepsilon_1} & b_i & a_{s_1}^{\varepsilon_1} a_{s_2}^{\varepsilon_2} \dots a_{s_L}^{\varepsilon_L} & \end{array}$$

This conjugating sequence is the same for each b_i and is defined by the private key a . The main goal of the attack is to reverse the sequence (25), and going back from the bottom to the top recover each conjugating factor. If successful, the procedure will result in the actual conjugator as a product of generators of A .

The next algorithm is the simplest realization of LBA called *the fastest descent LBA*. It takes as an input three tuples (a_1, \dots, a_m) , (b_1, \dots, b_n) , and (c_1, \dots, c_n) , where the last tuple is assumed to be $\bar{b}_1^a, \dots, \bar{b}_n^a$. The algorithm is a sequence of the following steps:

- **(Initialization)** Put $x = \varepsilon$ (where ε is the identity).
- **(Main loop)** For each $i = 1, \dots, m$ and $\varepsilon = \pm 1$ compute $l_{i,\varepsilon} = \sum_{j=1}^n l_X(a_i^{-\varepsilon} c_j a_i^\varepsilon)$.
 - If for each $i = 1, \dots, n$ and $\varepsilon = \pm 1$ the inequality $l_{i,\varepsilon} > \sum_{j=1}^n l_X(c_j)$ is satisfied, then output x .
 - Otherwise pick i and ε giving the smallest value of $l_{i,\varepsilon}$. Multiply x on the right by a_i^ε . For each $j = 1, \dots, n$ conjugate $c_j = a_i^{-\varepsilon} c_j a_i^\varepsilon$. Continue.

- **(Last step)** If $c_j = b_j$ for each $j = 1, \dots, n$, then output the obtained element x . Otherwise output *Failure*.

Other variations of LBA suggested in [211] are “LBA with backtracking” and “generalized LBA”. We refer to [211] for a detailed discussion on these.

We note that instead of the length function l_X , one can use any other objective function satisfying assumptions (L1) and (L2). Here besides l_X we analyze the behavior of modifications of LBA relative to the following functions:

- (M1)** Instead of computing the geodesic length $l_X(v_i)$ of the element $v_i \in G$ compute the geodesic length $l_Z(v_i)$ in the subgroup H generated by $Z = \{u\} \cup Y$ (clearly, $v_i \in H$). In this case, LBA in G is reduced to LBA in H , which might be easier. We term l_Z the *inner* length in LBA.
- (M2)** It might be difficult to compute the lengths $l_X(w)$ or $l_Z(w)$. In this case, one can try to compute some “good”, say linear, approximations of $l_X(w)$ or $l_Z(w)$, and then use some heuristic algorithms to carry over LBA (see [211]).

These modifications can make LBA much more efficient. In what follows our main interest is in the generic-case time complexity of LBA. To formulate this precisely, one needs to describe the set of inputs for LBA and the corresponding distribution on them.

Recall that an input for the SCSP* in a given group G with a fixed finite generating set X consists of a finitely generated subgroup $A = \langle a_1, \dots, a_k \rangle$ of G given by a k -tuple $(a_1, \dots, a_k) \in F(X)^k$, and a finite system of conjugacy equations $u_i^x = v_i$, where $u_i, v_i \in F(X)$, $i = 1, \dots, m$, that has a solution in A . We denote this data by $\alpha = (T, b)$, where $T = (a_1, \dots, a_k, u_1, \dots, u_m)$ and $b = (v_1, \dots, v_m)$. The distinction that we make here between T and b will be used later on. For fixed positive integers m, k we denote the set of all inputs $\alpha = (T, b)$ as above by $I_{k,m}$.

The standard procedure to generate a “random” input of this type in the AAG protocol is as follows.

Random generation of inputs for LBA in a given G :

- pick a random $k \in \mathbb{N}$ from a fixed interval $K_0 \leq k \leq K_1$;
- pick randomly k words $a_1, \dots, a_k \in F(X)$ with the length in a fixed interval $L_0 \leq |a_i| \leq L_1$;
- pick a random $m \in \mathbb{N}$ from a fixed interval $M_0 \leq m \leq M_1$;
- pick randomly m words $u_1, \dots, u_m \in F(X)$ with the length in a fixed interval $N_0 \leq |u_i| \leq N_1$;
- pick a random element w from the subgroup $A = \langle a_1, \dots, a_k \rangle$, as a random product $w = a_{i_1} a_{i_2} \dots a_{i_c}$ of elements from $\{a_1, \dots, a_k\}$ with the number of factors c in a fixed interval $P_1 \leq c \leq P_2$;
- conjugate $v_i = u_i^w$ and compute the normal form \tilde{v}_i of v_i , $i = 1, \dots, m$.

As we have argued in Section 14.2, one can fix the numbers k, m and the number of factors c in the product w in advance. Observe that the choice of the elements v_1, \dots, v_m is completely determined by the choice of the tuple $T = (a_1, \dots, a_k, u_1, \dots, u_m) \in F(X)^{k+m}$ and the word w .

Note also that the distribution on the subgroups $H = \langle T \rangle$ (more precisely, on their descriptions from $F(X)^{k+m}$) that comes from the random generation procedure above coincides with the distribution on the $(k+m)$ -generated subgroups that

was described in Section 14.2. We summarize these observations in the following remark.

REMARK 15.2.1.

- 1) The choice of a tuple $T = (a_1, \dots, a_k, u_1, \dots, u_m) \in F(X)^{k+m}$ precisely corresponds to the choice of generators of random subgroups described in Section 14.2.
- 2) Asymptotic properties of subgroups generated by T precisely correspond to the asymptotic properties of subgroups discussed in Section 14.

15.2.2. LBA in free groups. In this section we discuss LBA in free groups. Note that there are fast (quadratic time) algorithms to solve the SCSP*, and hence the AAG problem, in free groups (see Section 15.4.2). However, results on LBA in free groups will serve as a base for us to solve the SCSP* in many other groups.

Let k be a fixed positive natural number. We say that cancellation in a set of words $Y = \{y_1, \dots, y_k\} \subseteq F(X)^k$ is at most λ , where $\lambda \in (0, 1/2)$, if for any $u, v \in Y^{\pm 1}$ the amount of cancellation in the product uv is strictly less than $\lambda \min\{l_X(u), l_X(v)\}$, provided $u \neq v^{-1}$ in $F(X)$.

We summarize a couple of well-known facts in the following lemma.

LEMMA 15.2.2. *If the set $Y = \{y_1, \dots, y_k\}$ satisfies the λ -condition for some $\lambda \in (0, 1/2)$, then:*

- *The set Y is Nielsen reduced. In particular, Y freely generates a free subgroup and any element $w \in \langle Y \rangle$ can be uniquely represented as a reduced word in the generators Y and their inverses.*
- *The membership search problem for a subgroup $\langle Y \rangle$ is decidable in linear time.*
- *The geodesic length for elements of a subgroup $\langle Y \rangle$ is computable in linear time.*

The following result was proved in [185].

THEOREM 15.2.3. *Let $\lambda \in (0, 1/2)$. The set S of k -tuples $(u_1, \dots, u_k) \in F(X)^k$ satisfying the λ -condition is exponentially generic, and hence the set of k -tuples that are the Nielsen reduced in $F(X)$ is exponentially generic, too.*

Now we are ready to discuss the generic-case complexity of LBA in free groups.

THEOREM 15.2.4. *Let $F(X)$ be the free group with basis X . Let $v_1 = u_1^x, \dots, v_n = u_n^x$ be a system of conjugacy equations in $F(X)$, where x is searched in the subgroup A generated by a_1, \dots, a_m . Let Z be the tuple $(u_1, \dots, u_n, a_1, \dots, a_m)$. Then LBA with respect to the length function l_Z solves the SCSP* in linear time on an exponentially generic set of inputs.*

Proof. Let n and m be fixed positive integers. Denote by S the set of $(n+m)$ -tuples $(u_1, \dots, u_n, a_1, \dots, a_m) \in F(X)^{n+m}$ that satisfy the $1/4$ -condition. It follows from Theorem 15.2.3 that the set S is exponentially generic.

Furthermore, the system of conjugacy equations associated with such a tuple $Z = (u_1, \dots, u_n, a_1, \dots, a_m)$ has the form

$$\begin{cases} v_1 = u_1^x \\ \dots \\ v_n = u_n^x, \end{cases}$$

where v_i belong to the subgroup $H = \langle Z \rangle$ generated by Z , and x is searched in the same subgroup. By Lemma 15.2.2 one can find expressions for v_i in terms of the generators Z in linear time. Now, since the generators a_1, \dots, a_m are part of the basis of the subgroup H , it follows that LBA relative to l_Z successfully finds a solution $x = w(a_1, \dots, a_m)$ in linear time. ■

15.2.3. LBA in groups from \mathcal{FB}_{exp} . The results of the previous section are not very surprising because of the nature of cancellation in free groups. What does look surprising is that LBA works generically in some other groups that seem to be “far” from free groups. In this and the next section we outline a general mathematical explanation of this phenomenon. In particular, it will be clear why the modification (M1) of LBA, which was discussed in Section 15.2.1, is very robust, provided one can compute the geodesic length in subgroups.

We start with a slight generalization of Theorem 15.2.4. Recall (from Section 15.2.1) that inputs for LBA, as well as for the SCSP*, can be described in the form $\alpha = (T, b)$, where $T = (a_1, \dots, a_k, u_1, \dots, u_m) \in F(X)^{k+m}$ and $b = (v_1, \dots, v_m)$, such that there is a solution of the system $u_i^x = v_i$ in the subgroup $A = \langle a_1, \dots, a_k \rangle$.

LEMMA 15.2.5. *Let G be a group with a finite generating set X , and $I_{k,m}$ the set of all inputs (T, b) for LBA in G . Put*

$$I_{free} = \{(T, b) \in I_{k,m} \mid T \text{ freely generates a free subgroup in } G\}.$$

Suppose there is an exponentially generic subset S of I_{free} and an algorithm \mathcal{A} that computes the geodesic length l_T of elements from the subgroup $\langle T \rangle$, $(T, b) \in S$, when these elements are given as words from $F(X)$. Then there is an exponentially generic subset S' of I_{free} such that LBA halts on inputs from S' and outputs a solution for the related SCSP in at most quadratic time modulo the algorithm \mathcal{A} .*

Proof. The result follows directly follows from Theorem 15.2.4. ■

Let $G \in \mathcal{FB}_{exp}$. In the next theorem we prove that time complexity of the SCSP* on an exponentially generic set of inputs is at most quadratic modulo time complexity of the problem of computing the geodesic length in finitely generated subgroup of G .

THEOREM 15.2.6 (Reducibility to subgroup-length function). *Let G be a group with exponentially generic free basis property, and X a finite generating set of G . Then there is an exponentially generic subset S of the set $I_{k,m}$ of all inputs for LBA in G such that LBA relative to l_T halts on inputs from S and outputs a solution for the related SCSP*. Moreover, time complexity of LBA on inputs from S is at most quadratic modulo the algorithm \mathcal{A} that computes the geodesic length l_T of elements from the subgroup $\langle T \rangle$ when these elements are given as words from $F(X)$.*

Proof. By Lemma 15.2.5 there is an exponentially generic subset S of I_{free} such that LBA halts on inputs from S and outputs a solution for the related SCSP*. Moreover, time complexity of LBA on inputs from S is at most quadratic modulo the algorithm \mathcal{A} that computes the geodesic length l_T of elements from the subgroup $\langle T \rangle$ when these elements are given as words from $F(X)$. It suffices to show now that the set I_{free} is exponentially generic in the set of all inputs I for LBA in G . By Remark 15.2.1, the asymptotic density of the set I_{free} in I is the same as the asymptotic density of the set of tuples $T \in F(X)^{k+m}$ which have free basis property in G . Since G is in \mathcal{FB}_{exp} , this set is exponentially generic in $F(X)^{k+m}$, so is I_{free} in I . This proves the theorem. ■

15.3. Computing the geodesic length in a subgroup

For groups $G \in \mathcal{FB}_{exp}$, Theorem 15.2.6 reduces (in quadratic time) time complexity of LBA on an exponentially generic set of inputs to time complexity of the problem of computing the geodesic length in finitely generated subgroups of G . In this section we discuss time complexity of algorithms to compute the geodesic length in a subgroup of G . This discussion is related to the modification (M2) of LBA introduced in Section 15.2.1. In particular, we focus on the situation where we do not have fast algorithms to compute the geodesic length of elements in finitely generated subgroups of G , or even in the group G itself. In this case, as was mentioned in the modification (M2), one can try to compute some linear approximations of these lengths and then use heuristic algorithms to carry out LBA.

In Section 15.3.2 we discuss hardness of the problem of computing the geodesic length (GLP) in braid groups B_n , the original platforms of the AAG protocol. Time complexity of the GLP in B_n relative to the standard set Σ of Artin generators is unknown. We discuss some recent results and conjectures in this area. However, there are efficient linear approximations of the geodesic length in B_n relative to the set Δ of generators (the set of generalized half-twists). Theoretically, this gives linear approximations of the geodesic length of elements in B_n in the Artin generators and, furthermore, linear approximations of the geodesic inner length in quasi-isometrically embedded subgroups. If, as conjectured, the set of quasi-isometrically embedded subgroups is exponentially generic in braid groups, then this gives a sound foundation for LBA in braid groups. Note that linear approximations alone are not quite sufficient for successful LBA. To get a precise solution of the SCSP*, one also needs a robust “local search” near a given approximation of the solution. To this end several efficient heuristic algorithms have been developed; see e.g. [192], [211]. However, so far none of them exploited directly the interesting interplay between geodesic lengths in Σ and Δ , as well as quasi-isometric embeddings of subgroups.

15.3.1. Related algorithmic problems. We start with precise formulation of some problems related to computing geodesics in G .

Computing the geodesic length in a group (GLP): Let G be a group with a finite generating set X . Given an element $w \in G$ as a product of generators from X , compute the geodesic length $l_X(w)$.

Computing the geodesic length in a subgroup (GLSP): Let G be a group with a finite generating set X , and A a subgroup of G generated by a finite set of elements $Y = \{a_1, \dots, a_k\}$ of G given as words from $F(X)$. Given an element $w \in A$ as a product of generators of A , compute the geodesic length $l_Y(w)$.

There is another (harder) variation of this problem that comes from the SCSP* problem:

Computing the geodesic length in a subgroup (GLSP*): Let G be a group with a finite generating set X , and A a subgroup of G generated by a finite set of elements $Y = \{a_1, \dots, a_k\}$ of G given as words from $F(X)$. Given an element $w \in A$ as a word from $F(X)$, compute the geodesic length $l_Y(w)$.

The following lemma is obvious. Recall that the membership search problem (MSP) for a subgroup A in G asks for a given element $w \in F(X)$, which belongs to A , to find a decomposition of w into a product of generators from Y and their inverses.

LEMMA 15.3.1. *Let G be a finitely generated group and A a finitely generated subgroup of G . Then:*

- 1) *The GLSP is linear time reducible to the GLSP*;*
- 2) *The GLSP* is linear time reducible to the GLSP modulo the MSP in A .*

Observe that if the GLSP has a fast solution for $A = G$ in G , then there is a fast algorithm to find the geodesic length of elements of G with respect to X . In particular, the word problem in G has a fast decision algorithm. In some groups, like free groups or partially commutative groups, given by standard generating sets, there are fast algorithms for computing the geodesic length of elements. In many other groups, like braid groups, or nilpotent groups, the computation of the geodesic length of elements is hard. Nevertheless, in many applications, including cryptography, it suffices to have a fast algorithm to compute a reasonable, say linear, approximation of the geodesic length of a given element. This motivates the following problem.

Computing a linear approximation of the geodesic length in a group (AGL): Let G be a group with a finite generating set X . Given a word $w \in F(X)$ compute a linear approximation of the geodesic length of w . More precisely, find an algorithm that for $w \in F(X)$ outputs a word $w' \in F(X)$ such that $\lambda l_X(w) + c \geq l_X(w')$, where λ and c are independent of w .

Another problem is to compute a good approximation in a subgroup of a group.

Computing a linear approximation of the geodesic length in a subgroup (AGLS): Let G be a group with a finite generating set X , and A a subgroup of G generated by a finite set of elements $Y = \{a_1, \dots, a_k\}$ of G given as words from $F(X)$. Given an element $w \in A$ as a word from $F(X)$, compute a linear approximation of the geodesic length $l_Y(w)$ of w .

Assume now that there is a fast algorithm to solve the AGL problem in the group G . This does not imply that there is a fast algorithm to compute a linear approximation of the geodesic length in a given subgroup A of G , unless the subgroup A is quasi-isometrically embedded in G .

LEMMA 15.3.2. *Let G be a group with a finite generating set X , and \mathcal{A} an algorithm that solves the AGL problem in G with respect to X . If H is a quasi-isometrically embedded subgroup of G generated by a finite set Y , then for every $w \in H$ given as a word from $F(X)$, the algorithm \mathcal{A} outputs a word $w' \in F(X)$ such that $l_Y(w) \leq \mu l_X(w') + d$ for some constants μ and d , which depend only on \mathcal{A} and H .*

Proof. The proof is straightforward. ■

15.3.2. Geodesic length in braid groups. There are no known efficient algorithms to compute the geodesic length of elements in braid groups with respect to the set Σ of the standard Artin's generators. Some indications that this could be a hard problem are given in [221], where the authors prove that the set of geodesics in B_∞ is co-NP-complete. However, in a given group, the problem of computing the length of a word could be easier than the problem of finding a geodesic of the word. Moreover, complexity of a set of geodesics in a group may not be a good indicator of the time complexity of computing the geodesic length in a randomly chosen subgroup. In fact, it has been shown in [192, 193] that in a braid group B_n one can efficiently compute a reasonable approximation of the length function on B_n (relative to Σ), which gives a foundation for successful LBA, without computing the length in the group. Furthermore, there are interesting conjectures that, if settled affirmatively, will lead to more efficient algorithms for computing the length of elements in braid groups and their subgroups. To explain this we need to first recall some known facts and terminology.

We remind the reader that the group B_n has the standard Artin's presentation:

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{ll} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j & \text{if } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i & \text{if } |i - j| > 1 \end{array} \right\rangle.$$

We denote the corresponding generating set $\{\sigma_1, \dots, \sigma_{n-1}\}$ by Σ and the corresponding length function by $l_\Sigma(w)$.

Elements in B_n admit so-called *Garside normal forms* (cf. our Section 5.1.3 in Chapter 5). These forms are unique and the time complexity to compute the Garside normal form of an element of B_n given by a word $w \in F(\Sigma)$ is bounded by $O(|w|^2 n^2)$. However, Garside normal forms are far from being geodesic in B_n .

In 1991 Dehornoy introduced [54] the following notion of σ -positive braid word and a handle-reduction algorithm to compute a σ -positive representative of a given word. A braid word w is termed to be σ_k -positive (respectively, σ_k -negative), if it contains σ_k , but does not contain σ_k^{-1} or $\sigma_i^{\pm 1}$ with $i < k$ (respectively, contains σ_k^{-1} , but not σ_k or $\sigma_i^{\pm 1}$ with $i < k$). A braid word w is said to be σ -positive (respectively, σ -negative), if it is σ_k -positive (respectively, σ_k -negative) for some $k \leq n - 1$. A braid word w is said to be σ -consistent if it is either trivial, or σ -positive, or σ -negative.

THEOREM 15.3.3 (Dehornoy [54]). *For any braid $\beta \in B_n$, exactly one of the following is true:*

- 1) β is trivial;
- 2) β can be presented by a σ_k -positive braid word for some k ;
- 3) β can be presented by a σ_k -negative braid word for some k .

Thus, it makes sense to speak about σ -positive and σ_k -positive (or σ -, σ_k -negative) braids.

The following question is of primary interest when solving the AGL problem in braid groups: is there a polynomial $p(x)$ such that for every word $w \in F(\Sigma)$, $p(l_\Sigma(w))$ gives an upper bound for the Σ -length of the shortest σ -consistent braid word representing $w \in B_n$? Dehornoy's original algorithms in [54], as well as the handle reduction from [55]) and the algorithm from [77], give an exponential bound on the length of the shortest σ -consistent representative.

In [63] (see also [55, 77] for a related discussion) Dynnikov and Wiest formulated the following conjecture.

CONJECTURE 15.3.4. There are numbers λ, c such that every braid $w \in B_n$ has a σ -consistent representative whose Σ -length is bounded linearly by the Σ -length of the braid.

They also showed that this conjecture holds if the Σ -length of elements is replaced by the Δ -length (relative to the set Δ of generators).

The set Δ of generators consists of the braids $\Delta_{ij}, 1 \leq i < j \leq n$, which are the half-twists of strands i through j :

$$\Delta_{ij} = (\sigma_i \dots \sigma_{j-1})(\sigma_i \dots \sigma_{j-2}) \dots \sigma_i.$$

Δ is a generating set of B_n , containing the Artins generators $\sigma_i = \Delta_{i,i+1}$ and the Garside fundamental braid Δ_{1n} . The *compressed* Δ -length of a word w of the form

$$w = \Delta_{i_1 j_1}^{k_1} \dots \Delta_{i_s j_s}^{k_s},$$

where $k_t \neq 0$ and $\Delta_{i_t, j_t} \neq \Delta_{i_{t+1}, j_{t+1}}$ for all t , is defined by

$$L_\Delta(w) = \sum_{i=1} \log_2(|k_i| + 1).$$

For an element $\beta \in B_n$, the value $L_\Delta(\beta)$ is defined by

$$L_\Delta(\beta) = \min\{L_\Delta(w) \mid \text{the word } w \text{ represents } \beta\}.$$

Obviously, for any braid β one has

$$L_\Delta(\beta) \leq l_\Delta(\beta) \leq l_\Sigma(\beta).$$

The modified conjecture assumes the following extension of the notion of a σ -positive braid word: a word in the alphabet $\Delta = \{\Delta_{ij} \mid 0 < i < j < n\}$ is said to be σ -positive if, for some $k < l$, it contains Δ_{kl} , and contains neither Δ_{kj}^{-1} nor $\Delta_{ij}^{\pm 1}$ with $i < k$ and any j . In other words, a word w in letters Δ_{ij} is σ -positive (negative) if the word in standard generators σ_i obtained from w by the obvious expansion is.

THEOREM 15.3.5 (Dynnikov, Wiest [63]). *Any braid $\beta \in B_n$ can be presented by a σ -consistent word w in the alphabet $\{\Delta_{ij}\}$ such that*

$$l_\Delta(w) \leq 30nl_\Delta(\beta).$$

This theorem gives a method to approximate geodesic length in braid groups, as well as in their quasi-isometrically embedded subgroups. It remains to be seen whether or not this could lead to more efficient versions of LBA.

15.4. Quotient attacks

In this section we describe another type of attacks, called *quotient attacks* (QA). In fact, quotient attacks are just fast generic algorithms to solve some search problems in groups, such as the Membership Search Problem (MSP), the Simultaneous Conjugacy Search Problem (SCSP), the Simultaneous Conjugacy Search Problem relative a to a subgroup (SCSP*), etc. The main idea behind QA is that to solve a problem in a group G it suffices, on most inputs, to solve it in a quotient G/N , provided G/N has the generic free basis property and a fast decision algorithm for the relevant problem. In particular, this is the case if G has a free non-abelian quotient. Note that a similar idea was already exploited in [146], but there the answer was given only for inputs in the ‘‘No’’ part of the decision problem, which obviously is of no use for search problems. The strength of our approach comes from the extra requirement that G/N has the free basis property.

In Sections 15.4.1 and 15.4.2 we discuss the conjugacy and membership problems (in all their variations) in free groups. Some of these results were known as “folklore”, some could be found in the literature. Nevertheless, we sketch most of the proofs here, since this will serve us as the ground for solving similar problems in other groups.

15.4.1. Membership problems in free groups. In this section we discuss some algorithms to solve different versions of the membership problem in free groups. We start by reminding the classical membership problem (MP). Everywhere below G is a fixed group generated by a finite set X .

The Membership Problem (MP): Let $A = \langle a_1, \dots, a_m \rangle$ be a fixed finitely generated subgroup of G given by a finite set of generators a_1, \dots, a_m (viewed as words in $F(X)$). Given a word $w \in F(X)$, decide whether or not w belongs to A .

When the subgroup A is not fixed, but comes as part of the input (as in AAG scheme), the problem is more adequately described in its *uniform* version.

The Uniform Membership Problem (UMP): Given a finite tuple of elements $w, a_1, \dots, a_m \in F(X)$, decide whether or not w (viewed as an element of G) belongs to the subgroup A generated by the elements a_1, \dots, a_m in G .

To solve the MP in free groups, we use the folding technique introduced by Stallings [263] (see also [145] for a more detailed treatment). Given a tuple of words $a_1, \dots, a_m \in F(X)$ one can construct a finite deterministic automaton Γ_A , which accepts a reduced word $w \in F(X)$ if and only if w belongs to the subgroup $A = \langle a_1, \dots, a_m \rangle$ generated by a_1, \dots, a_m in $F(X)$.

To determine the time complexity of the MP and UMP, recall that for a given positive integer n , the function $\log^* n$ is defined as the least natural number m such that the m -tower of exponents of 2 exceeds n , or equivalently, $\log_2 \circ \log_2 \circ \dots \circ \log_2(n) \leq 1$, where on the left one has the composition of m logarithms.

LEMMA 15.4.1. *There is an algorithm that for any input $w, a_1, \dots, a_m \in F(X)$ for the UMP finds the correct answer in nearly linear time $O(|w| + n \log^* n)$, where $n = \sum_{i=1}^k |a_i|$. Furthermore, the algorithm works in linear time $O(|w| + n)$ on an exponentially generic set of inputs.*

Proof. Indeed, given $w, a_1, \dots, a_m \in F(X)$ one can construct Γ_A in time $O(n \log^* n)$ (see [271]) and check whether or not Γ_A accepts w in time $O(|w|)$, as required.

To prove the generic estimate recall that the set of m -tuples $a_1, \dots, a_m \in F(X)$ satisfying the $1/4$ -condition is exponentially generic, and Stallings’ procedure gives the automaton Γ_A in linear time $O(n)$. ■

In cryptography, the search versions of the MP and UMP are most interesting.

The Membership Search Problem (MSP): Let $A = \langle a_1, \dots, a_m \rangle$ be a fixed finitely generated subgroup of G given by a finite set of generators a_1, \dots, a_m , viewed as words in $F(X)$. Given a word $w \in F(X)$ that belongs to A , find a representation of w as a product of the generators a_1, \dots, a_m and their inverses.

The Uniform Membership Search Problem (UMSP): Given a finite set of elements $w, a_1, \dots, a_m \in F(X)$ such that $w \in A = \langle a_1, \dots, a_m \rangle$, find a representation of w as a product of the generators a_1, \dots, a_m and their inverses.

Upper bounds for the time complexity of the MSP follow easily from the corresponding bounds for the MP.

LEMMA 15.4.2. *The time complexity of the MSP in a free group is bounded from above by $O(|w|)$.*

Proof. Let $A = \langle a_1, \dots, a_m \rangle$ be a fixed finitely generated subgroup of G . As we mentioned above, one can construct Stallings' folding Γ_A in time $O(n \log^* n)$, where $n = |a_1| + \dots + |a_n|$. Then, one can construct, in linear time in n , a Nielsen basis $S = \{b_1, \dots, b_n\}$ for A by using the breadth-first search (see [145]). Now, given a word $w \in F(X)$ that belongs to A , one can follow the accepting path for w in Γ_A and rewrite w as a product of generators from S and their inverses. This requires linear time in $|w|$. It suffices to notice that the elements b_i can be expressed as fixed products of elements from the initial generating set of A , i.e., $b_i = u_i(a_1, \dots, a_n)$, $i = 1, \dots, m$. Therefore, any expression of w as a product of elements from $S^{\pm 1}$ can be rewritten in linear time into a product of the initial generators. ■

Note that in the proof above we used the fact that any product of new generators b_i and their inversions can be rewritten in linear time into a product of the old generators a_i and their inversions. This was because we assumed that one can rewrite the new generators b_i as products of the old generators a_i in a constant time. This is correct if the subgroup A is fixed. Otherwise, say in the UMSP, the assumption does not hold. It is not even clear whether one can do it in polynomial time or not. In fact, the time complexity of the UMSP is unknown. The following problem is of great interest in this area.

PROBLEM 15.4.3. Is time complexity of the UMSP in free groups polynomial?

However, the generic-case complexity of the UMSP in free groups is known.

LEMMA 15.4.4. *The generic-case time complexity of the UMSP in free groups is linear. More precisely, there is an exponentially generic subset $T \subseteq F(X)^n$ such that for every tuple $(w, a_1, \dots, a_m) \in F(X) \times T$ with $w \in \langle a_1, \dots, a_m \rangle$, one can express w as a product of a_1, \dots, a_m and their inverses in time $O(|w| + n)$, where $n = |a_1| + \dots + |a_n|$.*

Proof. First, note that if in the argument in the proof of Lemma 15.4.2 the initial set of generators a_1, \dots, a_m of a subgroup A satisfies the 1/4-condition, then the set of the new generators b_1, \dots, b_m coincides with the set of the initial generators (see [145] for details). Moreover, as we mentioned in the proof of Theorem 15.2.4, the set T of tuples $(a_1, \dots, a_m) \in F(X)^m$ satisfying the 1/4-condition is exponentially generic. Hence the argument from Lemma 15.4.2 proves the required upper bound for the UMSP on T . ■

15.4.2. Conjugacy problems in free groups. Now we turn to the conjugacy problems in free groups. Again, everywhere below G is a fixed group generated by a finite set X .

It is easy to see that the CP and CSP in free groups are decidable in at most quadratic time. It is quite tricky though to show that the CP and CSP are decidable in free groups in linear time. This result is based on the Knuth-Morris-Pratt substring searching algorithm [160]. Similarly, the *root search problem* (see below) is decidable in free groups in linear time.

The Root Search Problem (RSP): Given a word $w \in F(X)$, find a shortest word $u \in F(X)$ such that $w = u^n$ for some positive integer n .

Note that the RSP in free groups can be interpreted as a problem of finding a generator of the centralizer of a non-trivial element (in a free group, such a centralizer is cyclic).

THEOREM 15.4.5. *The Simultaneous Conjugacy Problem (SCP) and Simultaneous Conjugacy Search Problem (SCSP) in free groups are reducible in linear time to the CP, CSP, and RSP. In particular, it is decidable in linear time.*

Proof. We briefly outline an algorithm that simultaneously solves the problems SCP and SCSP in free groups, i.e., given a finite system of conjugacy equations

$$(26) \quad \begin{cases} u_1^x = v_1, \\ \dots \\ u_n^x = v_n, \end{cases}$$

the algorithm decides whether or not this system has a solution in a free group $F(X)$, and if so, finds a solution. Using the decision algorithm for the CP one can check whether or not there is an equation in (26) that does not have solutions in F . If there is such an equation, the whole system does not have solutions in F , and we are done. Otherwise, using the algorithm to solve the CSP in F , one can find a particular solution d_i for every equation $u_i^x = v_i$ in (26). In this case the set of all solutions of the equation $u_i^x = v_i$ coincides with the coset $C(u_i)d_i$ of the centralizer $C(u_i)$. Observe that using the decision algorithm for the RSP, one can find a generator (the root of u_i) of the centralizer $C(u_i)$ in F .

Consider now the first two equations in (26). The system

$$(27) \quad u_1^x = v_1, u_2^x = v_2$$

has a solution in $F(X)$ if and only if the intersection $V = C(u_1)d_1 \cap C(u_2)d_2$ is non-empty. In this case

$$V = C(u_1)d_1 \cap C(u_2)d_2 = (C(u_1) \cap C(u_2))d$$

for some $d \in F$.

If $[u_1, u_2] = 1$, then V , as the intersection of two cosets, is non-trivial if and only if the cosets coincide, i.e., $[u_1, d_1 d_2^{-1}] = 1$. This can be checked in linear time (since the word problem in $F(X)$ has a linear time solution). Therefore, in linear time we either check that the system (27), and hence the system (26), does not have any solutions, or we confirm that (27) is equivalent to one of the equations, so (26) is equivalent to its own subsystem, where the first equation is removed. In the latter case the induction finishes the proof.

If $[u_1, u_2] \neq 1$, then $C(u_1) \cap C(u_2) = 1$, so either $V = \emptyset$ or $V = \{d\}$. In both cases one can easily find all solutions of (26). Indeed, if $V = \emptyset$, then (26) does not have any solutions. If $V = \{d\}$, then d is the only potential solution of (26), and one can check whether or not d satisfies all other equations in (26) in linear time by the direct substitution.

Now the problem is to verify, in linear time, whether $V = \emptyset$ or not, which is equivalent to solving an equation

$$(28) \quad u_1^m d_1 = u_2^k d_2$$

for some integers m, k . By finding, in linear time, cyclically reduced decompositions of u_1 and u_2 , one can rewrite the equation (28) into an equivalent one of the form

$$(29) \quad w_2^{-k} c w_1^m = b,$$

where w_1, w_2 are cyclically reduced forms of u_1, u_2 , and either $w_2^{-1}c$ or cw_1 (or both) are reduced as written, and b does not begin with w_2^{-1} and does not end with w_1 . Again in linear time, one can find the maximal possible cancellation in $w_2^{-k}c$ and in cw_1 , and rewrite (29) in the form

$$(30) \quad w_2^{-k} \tilde{w}_1^s = \tilde{b},$$

where \tilde{w}_1 is a cyclic permutation of w_1 , and $|\tilde{b}| \leq |b| + |w_1|$. Note that two cyclically reduced periodic words w_2, \tilde{w}_1 either commute or do not have a common subword of length exceeding $|w_2| + |\tilde{w}_1|$. If they commute, then the equation (30) becomes a power equation, which is easy to solve. Otherwise, executing (in linear time) possible cancellation on the left-hand side of (30), one arrives to an equation of the form

$$(31) \quad w_2^{-r} e \tilde{w}_1^t = \tilde{b},$$

where there is no cancellation at all. This equation can be easily solved for r and t . This completes the proof. ■

As we have seen in the proof of Theorem 15.4.5, one of the main difficulties in solving the SCSP in groups lies in computing the intersection of two finitely generated subgroups or their cosets. Note that finitely generated subgroups of $F(X)$ are regular sets (accepted by their Stallings' automata). It is well known in the language theory that the intersection of two regular sets is again regular, and one can find an automaton accepting the intersection in at most quadratic time. This yields the following corollary.

COROLLARY 15.4.6. *The SCSP* in free groups is decidable in at most quadratic time.*

Proof. Recall from the proof of Theorem 15.4.5 that the algorithm solving a finite system of conjugacy equations in a free group either decides that there is no solution to the system, or produces a unique solution, or gives the whole solution set as a coset Cd of some centralizer C . In the first case, the corresponding SCSP* has no solutions in a given finitely generated subgroup A . In the second case, given a unique solution w of the system one can construct the automaton Γ_A that accepts A , and check whether w is in A or not (it takes time $n \log^* n$). In the third case, one needs to check if the intersection $Cd \cap A$ is empty or not; this can be done in at most quadratic time, as we have mentioned before. ■

Observe from the proof of Corollary 15.4.6 that the most time-consuming case in solving the SCSP* in free groups occurs when all the elements u_1, \dots, u_n in the system (26) commute pairwise. The set of such inputs for the SCSP* is obviously exponentially negligible. On the other hand, as we have shown in Theorem 15.2.4, LBA relative to l_T solves the SCSP* in linear time on an exponentially generic subset of a free group. Thus, generically, the SCSP* in free groups is decidable in linear time.

Since the AAG problem (see Section 15.1.2) is reducible in linear time to the SCSP* (Lemma 15.1.2), we have the following.

COROLLARY 15.4.7. *In an arbitrary free group F :*

- 1) *The AAG problem in F is decidable in at most quadratic time in the size of the input (i.e., in the size of the public information in the AAG scheme).*
- 2) *The AAG problem in F is decidable in linear time on an exponentially generic set of inputs.*

15.4.3. The MSP and SCSP* in groups with “good” quotients. In this section we discuss generic complexity of the Membership Search Problem (MSP) and the Simultaneous Conjugacy Search Problem relative to a subgroup (SCSP*) in groups that have “good” factors in \mathcal{FB}_{exp} .

Let G be a group generated by a finite set X , G/N a quotient of G , and $\varphi : G \rightarrow G/N$ the canonical epimorphism. Let $H = \langle u_1, \dots, u_k \rangle$ be a finitely generated subgroup of G . To solve the membership search problem for H , one can employ the following simple heuristic idea which we formulate as an algorithm.

ALGORITHM 15.4.8 (Heuristic solution to the MSP).

INPUT: A word $w = w(X)$ and generators $\{u_1, \dots, u_k\} \subset F(X)$ of a subgroup H .

OUTPUT: A representation $W(u_1, \dots, u_k)$ of w as an element of H or *Failure*.

COMPUTATIONS:

- A. Compute the generators $u_1^\varphi, \dots, u_k^\varphi$ of H^φ in G/N , where $\varphi : G \rightarrow G/N$ is the canonical epimorphism.
- B. Compute w^φ , solve the MSP for w^φ in H^φ , and find a representation $W(u_1^\varphi, \dots, u_k^\varphi)$ of w^φ as a product of the generators of $u_1^\varphi, \dots, u_k^\varphi$ and their inverses.
- C. Check if $W(u_1, \dots, u_k)$ is equal to w in G . If this is the case, then output W . Otherwise output *Failure*.

Observe that to run Algorithm 15.4.8, one needs to be able to solve the MSP in the quotient G/N (Step B) and to check the result in the original group (Step C), i.e., to solve the word problem in G . If these conditions are satisfied, then Algorithm 15.4.8 is a partial deterministic correct algorithm, i.e., if it gives an answer, it is correct. However, it is far from being obvious, even if the conditions are satisfied, that this algorithm can be robust in any interesting class of groups. The next theorem, which is the main result of this section, states that Algorithm 15.4.8 is very robust for groups from \mathcal{FB}_{exp} , with a few additional requirements.

THEOREM 15.4.9 (Reduction to a quotient). *Let G be a group generated by a finite set X , and with the word problem in a complexity class $C_1(n)$. Suppose G/N is a quotient of G such that:*

- 1) $G/N \in \mathcal{FB}_{exp}$.
- 2) *The canonical epimorphism $\varphi : G \rightarrow G/N$ is computable in time $C_2(n)$.*
- 3) *For every $k \in \mathbb{N}$, there is an algorithm \mathcal{A}_k in a complexity class $C_3(n)$, which solves the membership search problem in G/N for an exponentially generic set $M_k \subseteq F(X)^k$ of descriptions of k -generated subgroups in G/N .*

Then, for every k , Algorithm 15.4.8 solves the membership search problem on an exponentially generic set $T_k \subseteq F(X)^k$ of descriptions of k -generated subgroups in G . Furthermore, Algorithm 15.4.8 belongs to the complexity class $C_1(n) + C_2(n) + C_3(n)$.

Proof. We need to show that Algorithm 15.4.8 successfully halts on an exponentially generic set of tuples from $F(X)^k$. By the conditions of the theorem, the set

S_k of all k -tuples from $F(X)^k$ whose images in G/N freely generate free subgroups, is exponentially generic, as well as the set M_k of all tuples from $F(X)^k$ where the algorithm \mathcal{A}_k applies. Hence the intersection $T_k = S_k \cap M_k$ is exponentially generic in $F(X)^k$. We claim that Algorithm 15.4.8 applies to subgroups with descriptions from T_k . Indeed, the algorithm \mathcal{A}_k applies to subgroups generated by tuples $Y = (u_1, \dots, u_k)$ from T_k , so if $w^\varphi \in H^\varphi = \langle Y^\varphi \rangle$, then \mathcal{A}_k outputs a required representation $w^\varphi = W(Y^\varphi)$ in G/N . Note that H^φ is freely generated by Y^φ since $Y \in S_k$; therefore, φ is injective on H . It follows that $w = W(Y)$ in G , as required. This completes the proof. ■

Theorems 15.2.6 and 15.4.9 yield the following

COROLLARY 15.4.10. *Let G be as in Theorem 15.4.9. Then, for every $k, m > 0$, there is an algorithm $\mathcal{C}_{k,m}$ that solves the SCSP* on an exponentially generic subset of the set $I_{k,m}$ of all possible inputs. Furthermore, $\mathcal{C}_{k,m}$ belongs to the complexity class $n^2 + C_1(n) + C_2(n) + C_3(n)$.*

COROLLARY 15.4.11. *Let G be a group of pure braids PB_n , $n \geq 3$, or a non-abelian partially commutative group $G(\Gamma)$. Then, for every $k, m > 0$, there is an algorithm $\mathcal{C}_{k,m}$ that solves the SCSP* on an exponentially generic subset of the set $I_{k,m}$ of all possible inputs. Furthermore, $\mathcal{C}_{k,m}$ belongs to the complexity class $O(n^2)$.*

Proof. Recall that in any pure braid group and in any non-abelian partially commutative group the word problem can be solved by a quadratic time algorithm. Now the statement follows from Corollary 15.4.10 and Corollaries 14.4.3 and 14.4.5 in Chapter 14. ■

Part 5

Word and Conjugacy Search Problems in Groups

Decision problems are problems of the following nature: given a property \mathcal{P} and an object \mathcal{O} , find out whether or not the object \mathcal{O} has the property \mathcal{P} . On the other hand, search problems are of the following nature: given a property \mathcal{P} and an object \mathcal{O} with the property \mathcal{P} , find a proof (sometimes called a “witness”) of the fact that \mathcal{O} has the property \mathcal{P} . This is a substantial shift of paradigm, and in fact, studying search problems often gives rise to new research avenues in mathematics, very different from those prompted by addressing the corresponding decision problems. To give just a couple of examples from different areas of mathematics, we can mention (1) the isoperimetric function that can be used to measure the complexity of a proof that a given word is trivial in a given group; (2) Reidemeister moves that can be used to measure the complexity of a proof that two given knot diagrams are those of two isotopic knots; (3) elementary row (or column) operations on a matrix over a ring that may be used to measure the complexity of a proof that a given square matrix is invertible. With respect to the last example we note that, although a more straightforward proof would be just producing the inverse matrix, the proof (if it exists) by elementary row (or column) operations provides a useful *stratification* of the relevant search problem, which allows one to allocate a search problem to one of the established complexity classes (e.g. **P** or **NP**) by converting it to a decision problem; in this particular example the latter would be asking whether or not a given matrix is a product of at most k elementary matrices, for a sufficiently large k (depending on the size of a given matrix).

The objective of this part of the book is to address various search problems in group theory. We note that decision problems in group theory have been studied for over 100 years now, since Dehn put forward, in the beginning of the 20th century, the three famous decision problems now often referred to as *Dehn’s problems*: the word problem, the conjugacy problem, and the isomorphism problem. Later, some of these problems were generalized, and many other decision problems were raised; we refer to [158] for a survey.

On the other hand, search problems in group theory and their complexity started to attract attention relatively recently. Complexity of the word search problem in a finitely presented group is reflected by isoperimetric and isodiametric functions of a finite presentation of this group, as introduced in [119] and [95] in 1985–1991. More recently, complexity of the conjugacy search problem has received a lot of attention, after a seminal paper [5] offered a cryptographic key exchange protocol that relied in its security on the complexity of the conjugacy search problem in braid groups. Later on, there were other proposals of cryptographic primitives that relied on the complexity of other search problems, including the word search problem, the subgroup membership search problem, the isomorphism search problem, the decomposition search problem, etc.

In this part of the book, we are going to primarily focus on the complexity of the word and conjugacy search problems in some specific groups (e.g. in free solvable groups), as well as in groups given by random finite presentations.

CHAPTER 16

Word Search Problem

16.1. Introduction

In this chapter we study generic properties of van Kampen diagrams over finitely presented groups. For these groups we propose a new generically fast algorithm for the word search problem.

Let G be a group given by a finite presentation $G = \langle X; R \rangle$. The notion of generic time complexity of the word problem in G was introduced in [146]. A hierarchy of generic sets was developed in [31]. It allows one to compare sizes of various generic sets. We briefly describe some of these notions here. Let μ be a probability distribution on the set $(X^{\pm 1})^*$ of all words in the alphabet $X^{\pm 1} = X \cup X^{-1}$, or, more generally, an arbitrary additive function with values in $[0, 1]$ defined on some subsets of $(X^{\pm 1})^*$ (for instance, the asymptotic density on $(X^{\pm 1})^*$). A subset $T \subset (X^{\pm 1})^*$ is called *generic with respect to μ* (or μ -generic) if $\mu(T) = \mu(X)$. We say that the word problem for $G = \langle X; R \rangle$ has *polynomial-time generic-case complexity* with respect to μ if there exists an algorithm Ω and a μ -generic subset T of $(X^{\pm 1})^*$ such that Ω solves the word problem in polynomial time on all inputs from the set T . The generic-case complexity is in the spirit of but quite different from the average-case complexity in several respects (see [146] and [147] for details). One of the crucial differences is that the generic-case complexity is well-defined even for undecidable problems.

The main result of [146] states that the word problem is “generically fast” for a wide variety of finitely presented groups. The key idea there is very simple (though it is not very easy to provide a formal argument): if $\varphi : G \rightarrow H$ is a homomorphism of G onto, say, a hyperbolic group H , then the set $\{g \in G \mid g^\varphi \neq 1\}$ is generic in G with respect to the asymptotic density on G . Since the word problem in H has linear time complexity one can quickly verify the condition $g^\varphi \neq 1$, hence the result. Results of this type clarify the generic complexity of the “No” part of the word problem.

In most applications, however, one has to deal with the “Yes” part of a decision problem. For example, in modern cryptography security of various cryptosystems is based on presumably high generic-case complexity of the underlying search problems (see discussion in [194]), when the attacker knows that the private key does exit, the only problem is to find it in a feasible time.

In the algorithmic part of this chapter we focus on the generic-case time complexity of the word search problem in finitely presented groups. We describe a deterministic algorithm \mathcal{A}_W which for a given finite symmetric and *reduced* (a minor restriction, see Section 16.2) presentation $\langle X; R \rangle$ and a word w in the alphabet $X^{\pm 1}$ halts and returns the answer *Yes* if and only if w represents the trivial element

in the group $G = \langle X; R \rangle$. In this event some other algorithm \mathcal{B} gives a decomposition of w as a product of conjugates of relators from R in the compressed form of a *derivation* W . Moreover, if the presentation is symmetric and reduced then there is a generic subset T of $WP_{yes} = \{w \in (X \cup X^{-1})^* \mid w = 1 \text{ in } G\}$ and a polynomial $p(n)$ such that for a word $w \in T$ the algorithm \mathcal{A} (as well as the algorithm \mathcal{B}) gives the answer in at most $p_{\mathcal{A}}(|w|)$ (correspondingly, $p_{\mathcal{B}}(|w|)$) steps. We term a presentation *reduced* (see Section 16.2) if all the generators and all the proper subwords of the relators are not trivial in G . One can easily symmetrize a given presentation. Furthermore, every finite presentation $\langle X; R \rangle$ yields via a reduction (removing trivial generators and splitting relations when possible) a reduced presentation $\langle X'; R' \rangle$ such that $L(R') \leq L(R)$, where $L(R)$ is the total length of relations in R . If WP in the group $G = \langle X; R \rangle$ is decidable then the reduction process is effective, but in general it is not (otherwise one could test effectively triviality of the group given by a finite presentation). Luckily, many of the known presentations are reduced.

There are several key problems that one has to address when dealing with the “Yes” part of the word problem in a given group $G = \langle X; R \rangle$. First of all, it is the choice of the measure ν on WP_{yes} . Observe, that the set WP_{yes} is usually a negligible subset of $(X^{\pm 1})^*$ (the “No” part is big!), so the restriction onto WP_{yes} of any natural measure μ on $(X^{\pm 1})^*$ results in a set of measure zero, which is useless. This shows that the measure ν should be intrinsic to the nature of the trivial words in G . Here we elected to represent the trivial words in G by van Kampen diagrams of the finite presentation $\langle X; R \rangle$. At first glance the natural approach here would be to view the van Kampen diagram as a kind of graph and introduce the notion of a random diagram (hence a measure) by mimicking the classical random graph models. However, this immediately fails here (at least in its straightforward form) since the graphs corresponding to the van Kampen diagrams are planar, and for planar graphs the standard random graph models are not satisfactory. In view of this, we use the so-called *random generators* to describe random van Kampen diagrams. In Section 16.5 we discuss some natural random generators of various types and compare their properties. Our primary focus is on the *locally stable* generator RG_S described in Section 16.6. One of the most technical results of the chapter is Theorem 16.6.7 of Section 16.6.2 which states that the generator RG_S is *complete*, i.e., it generates (up to isomorphism) every van Kampen diagram with positive probability. Completeness of the other generators, which are not locally stable, follows much easier from their descriptions, but in this case it is harder to control their asymptotic properties.

The second fundamental problem that should be addressed here concerns the way one measures the “size” and “complexity” of trivial words. The obvious choice of the size function - the length of the input word - is not particularly good here, since it does not reflect the structure of the diagrams. Not surprisingly, there is no direct correlation between the length of the word and the “hardness” of the word problem. On the other hand, the area $\chi(D)$ of a diagram D (the total sum of faces and “free edges” in D) gives a more suitable estimation of the size of a diagram D . Still, there is no any direct relationship between the area of diagrams and the complexity of the classical decision word problem. For example, the Dehn’s function of the group $\langle x, y; x^{-1}yx = y^2 \rangle$ is exponential [96], but the time-complexity of the word problem is at most quadratic in the length of the input. The situation changes completely for the word search problem, where the Dehn’s function gives a

low bound for the *worst-case* time-complexity (if it is required to represent a given word from WP_{yes} as a product of conjugates of the relators). Notice, however, that Dehn's functions do not say much about average or generic-case complexities. One of the most interesting questions that arises here is: What is the isoperimetric inequality (the ratio of the area of the diagram to its perimeter) of a generic (random) diagram over a given presentation $\langle X; R \rangle$? If found, this “generic area” would provide a low bound for the *generic-case* complexity of the word search problem in $G = \langle X; R \rangle$. Now, another important question occurs: Is there an algorithm for the word search problem in $G = \langle X; R \rangle$ with the time function bounded by a polynomial in the area of the generic inputs? Observe, that knowledge of the area of a word w does not give immediately a fast algorithm to construct a van Kampen diagram for w (see Section 16.9). To address both of these questions we need a few definitions.

We believe that the real “algorithmic complexity” of a given diagram D (or the trivial word represented by D) depends on several geometric characteristics of D , including the area. In our approach the notion of *depth* of a diagram D plays the key part. Intuitively, the depth describes the minimal number of steps required to get from the boundary of D to the innermost cell of D , where a step is a move from a given cell c to any other cell having a common vertex with c . More precisely, the depth of a vertex $v \in D$ is a shortest sequence of vertices v_1, \dots, v_k , where $v_1 = v, v_k \in \partial D$ and for every pair of neighboring vertices v_i, v_{i+1} there is a cell c_i in D with $v_i, v_{i+1} \in \partial c_i$ (see Section 16.3.3 for details). Now the depth $\delta(D)$ is the maximum of the depth over all vertices in D . As usual, one can define *Depth* as filling function $\Delta : \mathbb{N} \rightarrow \mathbb{N}$ of a finite presentation $\langle X; R \rangle$. Namely, if $w \in WP_{yes}$ for $G = \langle X; R \rangle$ then define $\delta(w)$ to be equal to the minimal depth among all van Kampen diagrams representing w in $\langle X; R \rangle$, and put

$$\Delta(n) = \max\{\delta(w) \mid w \in WP_{yes} \text{ and } |w| \leq n\}.$$

The depth of diagrams allows one to estimate the time-complexity of the algorithms \mathcal{A} and \mathcal{B} above (see Section 16.4.2 for details). For a given word w the algorithm \mathcal{A} constructs finite *approximations* (singular subcomplexes) Γ of the Cayley graph $\Gamma(G, X)$ of G (see Section 16.3) until a closed path with the label w occurs in Γ . The principle idea behind this algorithm is that we do not enumerate all possible approximations Γ of $\Gamma(G, X)$ in some fixed computable order. Instead, the algorithm \mathcal{A} builds up approximations Γ layer by layer “around” the given word w inside $\Gamma(G, X)$, until the path with the label w becomes closed in the current approximation Γ . The resulting graph Γ (the 1-skeleton of the subcomplex) gives Stallings' folding (with respect to X) of some finitely generated subgroup of the normal closure of R in $F(X)$ which contains w . Now one can use the standard algorithms for the membership search problem in $F(X)$ to find a presentation of w as a product of conjugates of relators from R ; this is what the algorithm \mathcal{B} does. Notice, that the time complexity of the algorithm \mathcal{A} depends exponentially on the number of layers required to build up Γ from the segment labeled by w , which can be estimated from the above by $\delta(w)$, so the worst-time complexity of \mathcal{A} (as well as \mathcal{B}) is exponential in $\Delta(n)$. It is worthwhile to notice here that even though designing the algorithms \mathcal{A} and \mathcal{B} we were aiming at their generic performance in arbitrary presentations, they give worst-case fast (Ptime) decision and search algorithms for many presentations of interesting groups, in particular, of the hyperbolic

ones. Indeed, in Section 16.3.3 we show that a finitely generated group G is hyperbolic if and only if there is a constant $C > 0$ and a finite presentation $G = \langle X; R \rangle$ such that $\Delta(n) \leq C \log n$ for all $n \in \mathbb{N}$. For such presentations of hyperbolic groups the worst-case time complexity of the algorithms \mathcal{A} and \mathcal{B} is exponential in $\Delta(n)$, so it is polynomial in n . Notice, that in this case the algorithm \mathcal{A} solves WP in polynomial time in G even on the “No” part WP_{no} . Indeed, in this case it either halts on an input w and gives the answer “Yes” in polynomial time (for some fixed polynomial), or $w \neq 1$ in G .

In Section 16.9 we compare our algorithms \mathcal{A} and \mathcal{B} with several other known general algorithms for the word problem in groups, and argue that \mathcal{A} and \mathcal{B} generically outperform in the “Yes” part all known general algorithms for coset enumeration or enumeration of relators in finitely presented groups. To describe upper bounds on generic-case time-complexity of the algorithms \mathcal{A} and \mathcal{B} one has to study generic properties of van Kampen diagrams over $\langle X; R \rangle$. It turned out this topic is very interesting in its own right, and no doubt it needs some further research.

As we have mentioned already, to introduce a probability measure on diagrams we use some random generators (or *evolutionary models*). A diagram random generator RG builds up all diagrams over $\langle X; R \rangle$ (up to isomorphism) with positive probability starting with a single vertex and randomly (with prescribed probabilities) adding cells and edges, and randomly folding some of the edges with the same label. This process can be described as a simple random walk W on the corresponding infinite graph T (the transition graph of the random generator RG). The vertices of T are diagrams and two vertices are connected by a directed edge if one of them can be obtained from another by either adding an edge or a cell, or performing a fold. The standard Kolmogorov’s measure on the set of all trajectories of W induces a measure on the set of vertices of T . This allows one to introduce a measure on the set Ω of all isomorphism classes of van Kampen diagrams over $\langle X; R \rangle$ by adding up the probabilities assigned to the vertices in T which are isomorphic to a given diagram $D \in \Omega$.

The set Ω can be partitioned into subsets

$$\Omega_n = \{D \in \Omega \mid \chi(D) = n\}$$

consisting of all diagrams from Ω of area n . Now, we fix one of the random generators RG (RG_n or RG_S) described in Section 16.6 and denote by μ the probability measure on Ω associated with the random generator RG_S . We use the asymptotic density ρ on Ω (relative to the partition above and the measure μ) to measure asymptotic behavior of various subsets Q of Ω . As above,

$$\rho(Q) = \lim_{n \rightarrow \infty} \frac{\mu(Q \cap \Omega_n)}{\mu(\Omega_n)},$$

and one can define generic and exponentially generic sets Q .

Now we can give precise formulations of the main results of the paper. In Section 16.7 we prove (as Corollary 16.7.6) the following result which shows that generically the isoperimetric function on diagrams is linear (with the coefficient 4).

Theorem A. *The set of all van Kampen diagrams D over $\langle X; R \rangle$ with the length of the perimeter greater than $\frac{1}{4}\chi(D)$ is exponentially generic in Ω .*

This theorem gives a “local analog” of Gromov’s celebrated result that a random finitely presented group is hyperbolic. Indeed, according to the theorem, a

random van Kampen diagram over an arbitrary finite symmetrized reduced presentation is “hyperbolic”.

Furthermore, at the end of this section we show that generically the depth of diagrams from Ω is exponentially smaller than the area. Namely, we prove the following

Theorem B. *The set of all van Kampen diagrams D over $\langle X; R \rangle$ with the depth less than $\log \chi(D)$ is generic in Ω .*

Now, Theorem B allows one to estimate the generic time complexity of the algorithms \mathcal{A} and \mathcal{B} .

Theorem C. *The algorithms \mathcal{A} and \mathcal{B} are generically polynomial on WP_{yes} for every finite reduced presentation $\langle X; R \rangle$.*

Several interesting open problems arise here. The main concern is whether one can prove the results above for other natural distributions on van Kampen diagrams. The first problem is about van Kampen diagrams for reduced words in $X \cup X^{-1}$. Of course, this is a subclass of the set of diagrams we consider in this paper, but the induced distribution a priori might have different properties. Another problem is on generic properties of van Kampen diagrams which are uniformly distributed in the length of their perimeters. In this case evolutionary models are hard to apply, so one needs, it seems, quite different techniques.

16.2. Presentations of groups

Let X be a set. Denote by $X^{-1} = \{x^{-1} \mid x \in X\}$ the set of formal *inverses* of elements of X . The map $x \rightarrow x^{-1}$ ($x \in X$) naturally extends to an involution on the set $X^{\pm 1} = X \cup X^{-1}$, where we define $(x^{-1})^{-1} = x$. Let $M(X)$ be the *free monoid* with basis $X^{\pm 1}$ viewed as the set of all words in the alphabet $X^{\pm 1}$ with concatenation as the multiplication and $F = F(X)$ be a *free group* with basis X viewed as the set of all *reduced* words in $X^{\pm 1}$ with concatenation and subsequent reduction as the multiplication. For $R \subseteq F(X)$ and a group G we write

$$(32) \quad G = \langle X; R \rangle$$

if $G \simeq F(X)/gp_F(R)$, where $gp_F(R)$ is the normal closure of R in F . In this event X is a set of *generators* of G , R is a set of *relators* of G , and $\mathcal{P} = \langle X; R \rangle$ is a *presentation* of G . For a presentation $\langle X; R \rangle$ we define the total length $L(R)$ and the maximal length $M(R)$ of relators as

$$L(R) = \sum_{r \in R} |r|, \quad M(R) = \max_{r \in R} \{|r|\}.$$

A presentation $\mathcal{P} = \langle X; R \rangle$ is finite if the sets X and R are finite. A group G is called *finitely presented* if $G = \langle X; R \rangle$ for some finite presentation $\langle X; R \rangle$. In this chapter we discuss only finite presentations, though some results admit natural generalization for infinite presentations. A finite presentation $\langle X; R \rangle$ has *decidable word problem* if the set $gp_F(R)$ is recursive. It is not hard to see that if one finite presentation of G has decidable word problem then any finite presentation of G has decidable word problem. Therefore, we often refer to G as a group with decidable word problem.

For a subset $Y \subseteq F$ define $Y^{-1} = \{y^{-1} \mid y \in Y\}$.

DEFINITION 16.2.1. A set $R \subseteq F$ is called *symmetric* if the following hold:

- 1) every $r \in R$ is cyclically reduced;
- 2) $R^{-1} = R$;
- 3) if $r \in R$ then R contains every cyclic permutation of r .

For a given finite subset $R \subseteq F$ one can effectively construct a finite symmetric set R_{sym} such that $gp_F(R) = gp_F(R_{sym})$, so the groups defined by presentations $\langle X; R \rangle$ and $\langle X; R_{sym} \rangle$ are isomorphic. This allows one to consider only *symmetrized presentations*, i.e., presentations $\langle X; R \rangle$ with $R = R_{sym}$.

DEFINITION 16.2.2. A symmetrized presentation $G = \langle X; R \rangle$ is *G-reduced* if

- $x \neq_G 1$ for every $x \in X$;
- for every $r \in R$ and any $r_1, r_2 \in F(X) \setminus \{\varepsilon\}$ such that $r = r_1 \circ r_2$ we have

$$r_1, r_2 \neq_G 1.$$

PROPOSITION 16.2.3. If $G = \langle X; R \rangle$ is a finitely presented group with decidable word problem then one can effectively find a finite symmetrized *G-reduced* presentation of G .

Proof. If $\langle X; R \rangle$ is not reduced then we are in one of the following two cases.

- CASE 1. $x =_G 1$ for some $x \in X$. Then one can remove x from X and from every relator $r \in R$.
- CASE 2. For some $r \in R$ and $r_1, r_2 \in F(X) \setminus \{\varepsilon\}$ such that $r = r_1 \circ r_2$ we have $r_1, r_2 =_G 1$. In this case we split the relator r into two r_1 and r_2 and obtain a new set of relators

$$R' = (R \setminus \{r\}) \cup \{r_1, r_2\}.$$

The obtained presentation gives an isomorphic group.

In both cases the obtained presentation $\langle X'; R' \rangle$ defines an isomorphic group. If $\langle X'; R' \rangle$ is not symmetrized then we symmetrize it. It should be clear that this procedure simplifies the given presentation $\langle X; R \rangle$, and any sequence of such transformations stops in finitely many steps. The obtained group presentation is reduced, symmetrized, and defines the same group as the original presentation. ■

There are many known finite reduced presentations. For example, if $r \in F$ is a cyclically reduced word then one relator presentation $G = \langle X; r \rangle$ is *G-reduced* (this is due to Weinbaum [283], see also Theorem 5.29 in [177]); the standard presentation of the braid group B_n is B_n -reduced, as well as the canonical finite presentation of the Thompson group,

$$F = \langle x_0, x_1, x_2, x_3, x_4 \mid x_i^{-1} x_k x_i = x_{k+1} \ (k > i, k < 4) \rangle,$$

is reduced. In fact, most of the standard presentations of groups are reduced.

16.3. Approximating Cayley graphs of finitely presented groups

In this section we define the main objects of this paper such as van Kampen diagram, approximation of the Cayley graph, and *R-extension* procedure.

16.3.1. Cayley graph approximations and singular subcomplexes. Let

$$(33) \quad G = \langle X; R \rangle$$

be a symmetrized presentation of a group G . Recall that the Cayley graph $\Gamma(G, X)$ of G with respect to a generating set $X^{\pm 1}$ is a directed graph labeled by elements from $X^{\pm 1}$ (briefly *X-digraph*) such that elements of G form the vertex set of $\Gamma(G, X)$

and two vertices u and v are connected by a directed edge (u, v) (from u to v) with label $x \in X^{\pm 1}$ if and only if $v = ux$ in G . The edge (ux, u) is the inverse of (u, ux) , it has a label x^{-1} . One can turn $\Gamma(G, X)$ into a 2-dimensional *Cayley* complex $C(X, R)$ by adding a face for every loop in $\Gamma(G, X)$ with a label from R (see [177]).

An X -digraph Γ is called an *approximation* of the Cayley graph $\Gamma(G, X)$ of G if it comes equipped with an X -digraph morphism

$$\varphi : \Gamma \rightarrow \Gamma(G, X).$$

Each approximation (Γ, φ) of $\Gamma(G, X)$ gives rise to a 2-dimensional complex C_Γ over $\langle X; R \rangle$ (by adding cells for every closed path in Γ with labels from R) and a morphism of complexes (that preserves dimension, incidence, and labeling) $\varphi^* : C_\Gamma \rightarrow C(X, R)$. Such a pair (C_Γ, φ^*) is called a *singular subcomplex* of $C(X, R)$ (see [177]), or just an *R -complex*. We will freely switch from approximation graphs to the induced singular subcomplexes and back. Observe, that if Γ is connected then the map φ is unique up to the choice of vertices $v \in \Gamma$ and $v' \in \Gamma(G, X)$ such that $\varphi(v) = v'$.

In general, there is no any algorithm to check whether a given finite X -digraph Γ is an approximation of $\Gamma(G, X)$, or not. This problem is decidable if and only if the word problem in G is decidable. Below we describe a procedure to generate arbitrary large approximations of $\Gamma(G, X)$ with respect to a given presentation $\langle X; R \rangle$. This construction makes use of Stallings' *folding algorithm* (see [263]). For an X -digraph K by $S(K)$ we denote a *folded* X -digraph obtained from K by Stallings' folding procedure. The graph $S(K)$ is uniquely determined by K and there exists a canonical epimorphism $\varphi_S : K \rightarrow S(K)$ (see [263] and [145]). It is not hard to see that φ_S is a functor from the category of X -digraphs into the category of folded X -digraphs. For a graph Γ by $V(\Gamma)$ and $E(\Gamma)$ we denote, correspondingly, the sets of vertices and edges in Γ . The worst-case complexity for computing $S(K)$ is bounded from above by $O(|K| \log^* |K|)$ (see [271]), where $|K| = |V(K)| + |E(K)|$. The growth of \log^* is so slow, that in all practical computations can be viewed as bounded by a constant 5. In all practical computations the time complexity function is bounded by $O(|K|)$.

Recall, that the *core* of K is a subgraph $Core(K)$ of K formed by all closed cyclically reduced paths in K . If K has a fixed base point v then the *core* $Core_v(K)$ of K at v is formed by all closed reduced paths in K at the base point v . By definition K is a *core graph* (*core graph at v*) if $Core(K) = K$ ($Core_v(K) = K$).

Given a finite symmetrized presentation $\langle X; R \rangle$ and an arbitrary X -digraph K as an input, the procedure below outputs an X -digraph $\mathcal{C}(K)$ together with a morphism $\varphi_{\mathcal{C}} : K \rightarrow \mathcal{C}(K)$. We call $\mathcal{C}(K)$ the *R -extension* of K .

ALGORITHM 16.3.1. (R -extension of K).

INPUT: An X -digraph K .

OUTPUT: An X -digraph $\mathcal{C}(K)$ together with a morphism of X -digraphs $\varphi_{\mathcal{C}} : K \rightarrow \mathcal{C}(K)$.

COMPUTATIONS:

- C1) For each vertex $v \in K$ and each $r \in R$ add a cycle labeled by r to v . Denote the resulting graph by $\mathcal{C}_1(K)$ and the canonical embedding by $\varphi_{\mathcal{C}_1} : K \rightarrow \mathcal{C}_1(K)$.
- C2) Apply Stallings' procedure to fold the graph $\mathcal{C}_1(K)$. Put $\mathcal{C}(K) = S(\mathcal{C}_1(K))$ and $\varphi_{\mathcal{C}} = \varphi_S \circ \varphi_{\mathcal{C}_1}$.

C3) Output $\mathcal{C}(K)$ together with the morphism $\varphi_{\mathcal{C}} : K \rightarrow \mathcal{C}(K)$.

REMARK 16.3.2. If K has a distinguished based point v then we view $\mathcal{C}(K)$ as a graph with the distinguished based point $\varphi_{\mathcal{C}}(v)$.

LEMMA 16.3.3. *Let K be an X -digraph. Then the following holds:*

- (1) $\mathcal{C}(K)$ and $\mathcal{C}_1(K)$ are well-defined, i.e., do not depend on a sequence of actual transformations at steps C1) and C2) in Algorithm 16.3.1.
- (2) If $L(R) > 0$ then $\mathcal{C}_1(K)$ and $\mathcal{C}(K)$ are core graphs.
- (3) K is an approximation of $\Gamma(G, X)$ if and only if $\mathcal{C}(K)$ is an approximation of $\Gamma(G, X)$.
- (4) If $L(R) > 0$ then $\varphi_{\mathcal{C}}$ is a functor from the category of X -digraphs into the category of folded core X -digraphs.

Proof. (1) A graph $\mathcal{C}_1(K)$ does not depend on the order in which new loops are attached to K . Therefore, $\mathcal{C}_1(K)$ is well-defined. Since $\mathcal{C}(K)$ is obtained from $\mathcal{C}_1(K)$ by Stallings' folding procedure $\mathcal{C}(K)$ is well-defined too.

(2) Observe that after performing step C1) at every vertex $v \in \mathcal{C}_1(K)$ there is a loop labeled with some $r \in R_{sym}$. Hence $\mathcal{C}_1(K)$ is a core graph. Since $\varphi_S : \mathcal{C}_1(K) \rightarrow \mathcal{C}(K)$ is an epimorphism the same is true for any vertex of $\mathcal{C}(K)$.

(3) First, we show that K is an approximation of $\Gamma(G, X)$ if and only if $\mathcal{C}_1(K)$ is. Since K is a proper subgraph of $\mathcal{C}_1(K)$ the sufficiency is obvious. Assume that K is an approximation of $\Gamma(G, X)$ and $\varphi : K \rightarrow \Gamma(G, X)$ is an X -digraph morphism. Since $\mathcal{C}_1(K)$ is obtained from K by attaching a number of loops labeled by elements of R the morphism φ can be extended to $\mathcal{C}_1(K)$.

Finally, since taking Stallings' folding of a graph is a functor and $\Gamma(G, X)$ is a folded X -digraph it follows that $\mathcal{C}_1(K)$ is an approximation of $\Gamma(G, X)$ if and only if $\mathcal{C}(K)$ is.

(4) It follows from the definition of \mathcal{C}_1 that any morphism $\varphi : K \rightarrow L$ can be extended to a morphism $\psi : \mathcal{C}_1(K) \rightarrow \mathcal{C}_1(L)$ such that the following diagram commutes:

$$\begin{array}{ccc} K & \hookrightarrow & \mathcal{C}_1(K) \\ \downarrow \varphi & & \downarrow \psi \\ L & \hookrightarrow & \mathcal{C}_1(L) \end{array}$$

It is easy to see that $\varphi_{\mathcal{C}_1}$ is a functor from the category of X -digraphs into itself. By definition $\varphi_{\mathcal{C}} = \varphi_S \circ \varphi_{\mathcal{C}_1}$. As we have mentioned above φ_S is a functor from the category of X -digraphs to the category of folded X -digraphs and by 2) $\mathcal{C}(K)$ is a core graph. Hence the result. ■

LEMMA 16.3.4. *Let $\langle X; R \rangle$ be a symmetric finite presentation and K a finite X -digraph. Then the following inequalities hold for the R -extension $\mathcal{C}(K)$ of K :*

- (1) $|V(\mathcal{C}(K))| \leq (L(R) - |R| + 1)|V(K)|$;
- (2) $|E(\mathcal{C}(K))| \leq L(R)|V(K)| + |E(K)|$;
- (3) $|\mathcal{C}(K)| \leq (2L(R) - |R| + 1)|V(K)| + |E(K)|$;
- (4) The time complexity of Algorithm 16.3.1 is bounded above by $O(L(R)|V(K)| \log^*(L(R)|V(K)|))$.

Proof. It is easy to see that $|V(\mathcal{C}_1(K))| \leq (L(R) - |R| + 1)|V(K)|$ and $|E(\mathcal{C}_1(K))| \leq L(R)|V(K)| + |E(K)|$. Since Stallings' folding procedure does not increase the

number of the vertices and edges we immediately see that (1) and (2) hold. Since $|\mathcal{C}(K)| = |V(\mathcal{C}(K))| + |E(\mathcal{C}(K))|$, the property (3) follows from the properties (1) and (2).

It is straightforward to construct $\mathcal{C}_1(K)$ and it takes $O(|\mathcal{C}_1(K)|)$ steps to do that. As we mentioned above it requires $O(|\Gamma| \log^* |\Gamma|)$ to fold a given X -digraph Γ . Therefore, the time complexity of the Algorithm 16.3.1 is bounded above by $O(L(R)|V(K)| \log^*(L(R)|V(K)|))$, as claimed. ■

From now on we assume that every letter from X occurs in R . Starting with an X -digraph K one can iterate the construction above. Put:

$$\mathcal{C}^{(1)}(K) = \mathcal{C}(K), \quad \mathcal{C}^{(m+1)}(K) = \mathcal{C}(\mathcal{C}^{(m)}(K))$$

Similarly, one can define $\mathcal{C}_1^{(m)}(K)$ as the result of m consecutive applications of the unary operation \mathcal{C}_1 starting at K . As a special case define

$$\mathcal{C}^{(0)}(K) = S(K), \quad \mathcal{C}_1^{(0)}(K) = K.$$

LEMMA 16.3.5. *Let K be an X -digraph. Then the following holds for any non-negative integers m, n :*

- (1) $\mathcal{C}^{(m)}(K) \simeq S(\mathcal{C}_1^{(m)}(K))$;
- (2) $\mathcal{C}^{(m+n)}(K) \simeq \mathcal{C}^{(n)}(\mathcal{C}^{(m)}(K))$;
- (3) any morphism of X -digraphs $\varphi : L \rightarrow \mathcal{C}^{(m)}(K)$ gives rise to a morphism $\mathcal{C}^{(n)}(L) \rightarrow \mathcal{C}^{(m+n)}(K)$.
- (4) Let K_0 be a graph consisting of a single vertex (and no edges). Then

$$\Gamma(G, X) \simeq \lim_{m \rightarrow \infty} \mathcal{C}^{(m)}(K_0)$$

where $\Gamma(G, X)$ is the Cayley graph of the group $G = \langle X; R \rangle$.

Proof. We prove (1) by induction on m . If $m = 1$ then there is nothing to prove. Suppose then that $\mathcal{C}^{(m-1)}(K) \simeq S(\mathcal{C}_1^{(m-1)}(K))$. Observe that the diagram below commutes for an arbitrary X -digraph Γ :

$$\begin{array}{ccc} \Gamma & \xrightarrow{\varphi_{\mathcal{C}_1}} & \mathcal{C}_1(\Gamma) \\ \downarrow \varphi_S & & \downarrow \varphi_S \\ S(\Gamma) & \xrightarrow{\varphi_{\mathcal{C}_1}} & \mathcal{C}_1(S(\Gamma)) \xrightarrow{\varphi_S} \mathcal{C}(\Gamma) \end{array}$$

In particular, it commutes for $\Gamma = \mathcal{C}_1^{(m-1)}(K)$, which implies (1).

Property (2) is obvious. Property (3) follows from (2) and the fact that the map $\varphi_{\mathcal{C}}$ is a functor (Lemma 16.3.3). To see that (4) holds, observe that the graphs $\Gamma(G, X)$ and $\lim_{m \rightarrow \infty} \mathcal{C}^{(m)}(K_0)$ are both regular folded X -digraphs which both accept (as deterministic automata with the natural base points) the same language - the normal closure of R in $F(X)$. Hence they are isomorphic as X -digraphs. ■

REMARK 16.3.6. The construction above of the Cayley graph $\Gamma(G, X)$ as the direct limit of the graphs $\mathcal{C}^{(m)}(K_0)$ can be viewed as a variation of the coset enumeration procedure as it is described in [177, section III.12].

16.3.2. van Kampen diagrams. Now we consider a special type of singular subcomplexes of $C(X, R)$, so-called *van Kampen diagrams* (or *diagrams*) over $\langle X; R \rangle$.

There are two slightly different types of diagrams: the ones that are not necessarily homeomorphic to a Euclidean disc (introduced by Lyndon, see [177]) and the others which are always homeomorphic to a Euclidean disc (here, we refer to the Olshanskii's book [215]). We focus only on the diagrams of the first type, but a similar technique works for diagrams of the second type.

Let R^2 be the Euclidean plane. For a subset $S \subset R^2$ denote by ∂S the boundary of S , and by \overline{S} the closure of S in R^2 . Recall that a *map* \mathcal{M} is a finite disjoint union of *vertices* (points in R^2), edges (bounded subsets of R^2 homeomorphic to the open unit interval), and *faces* or *cells* (bounded sets homeomorphic to the open unit disc) which satisfies the following conditions:

- 1) if e is an edge in \mathcal{M} then there are two vertices $a, b \in \mathcal{M}$ (not necessary distinct) such that $\overline{e} = e \cup \{a\} \cup \{b\}$;
- 2) for each face $\Pi \in \mathcal{M}$ the boundary $\partial\Pi$ is connected and $\partial\Pi = \overline{e}_1 \cup \dots \cup \overline{e}_k$ for some edges $e_1, \dots, e_k \in \mathcal{M}$.

An edge of a map \mathcal{M} is called a *free edge* if it does not belong to the boundary of any cell in \mathcal{M} . By $V(\mathcal{M})$, $E(\mathcal{M})$, $FE(\mathcal{M})$, and $C(\mathcal{M})$ we denote the sets of vertices, edges, free edges, and cells in \mathcal{M} .

Now a *diagram*, or a *disc diagram* over a symmetrized presentation $\langle X; R \rangle$ is a connected and simply connected map \mathcal{M} such that:

- 1) every edge is considered as a pair of oppositely oriented edges e and e^{-1} ;
- 2) every oriented edge e is assigned a letter $y \in X^{\pm 1}$, which is called the *label* $\varphi(e)$ of e , and such that $\varphi(e^{-1}) = \varphi(e)^{-1}$. The function φ is called a *labeling* function;
- 3) if $p = e_1 \dots e_k$ is a *boundary cycle* or *contour* of a face Π (i.e., a cycle of minimal length containing all edges of $\partial\Pi$) then $\varphi(p) = \varphi(e_1) \dots \varphi(e_k) \in R$. In this case Π is called an *R-face*.

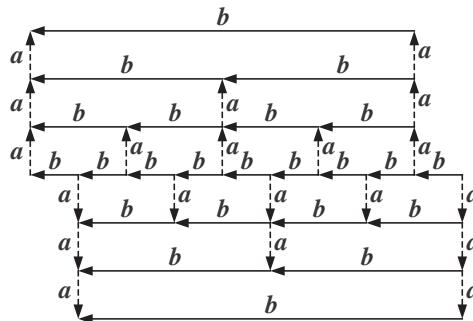


FIGURE 16.1. A diagram over a group $G = \langle a, b ; aba^{-1}b^{-2} \rangle$.

Let v be a vertex of a map \mathcal{M} . The *neighborhood* $N_{\mathcal{M}}(v)$ of v in \mathcal{M} is defined as the submap generated by all edges and faces in \mathcal{M} which are incident to v .

Let \mathcal{M} and \mathcal{N} be diagrams over $\langle X; R \rangle$. We say that \mathcal{M} and \mathcal{N} are *isomorphic* if there exists a homeomorphism of the Euclidean plane which induces an isomorphism

of corresponding 2-complexes. By $\text{Star}_{\mathcal{M}}(v)$ we denote the subgraph generated by all edges incident to v (including their endpoints). If presentation $\langle X; R \rangle$ is reduced and an orientation of the plane is fixed (clockwise or counterclockwise) then for any two edges $e_1, e_2 \in \text{Star}_{\mathcal{M}}(v)$ one can unambiguously define the set of all cells in $N_{\mathcal{M}}(v)$ and edges in $\text{Star}_{\mathcal{M}}(v)$ between e_1 and e_2 .

Diagrams over $\langle X; R \rangle$ satisfy the following important property.

LEMMA 16.3.7 (van Kampen Lemma, [177]). *Let $\langle X; R \rangle$ be a symmetrized presentation and w a word in the alphabet $X^{\pm 1}$. Then $w = 1$ in $G = \langle X; R \rangle$ if and only if there exists a diagram \mathcal{M} over $\langle X; R \rangle$ with a boundary label w .*

16.3.3. Depth of diagrams and the canonical embeddings.

DEFINITION 16.3.8. Let L be an R -complex. A sequence of vertices v_1, \dots, v_q in L is called a *vertex chain* if for any $i = 1, \dots, q - 1$ there is a cell or a free edge c such that $v_i, v_{i+1} \in \partial c$.

Also we will use the following version of chains.

DEFINITION 16.3.9. Let L be an R -complex. A sequence c_1, \dots, c_q in L , where c_i ($i = 1, \dots, q$) is a cell or a free edge, is called an *edge-cell chain* if for any $i = 1, \dots, q - 1$ we have

$$\partial c_i \cap \partial c_{i+1} \neq \emptyset.$$

Clearly, a vertex chain $v_1, \dots, v_q \in L$ defines at least one chain of cells and free edges $c_1, \dots, c_{q-1} \in L$ such that $v_i, v_{i+1} \in \partial c_i$ for $i = 1, \dots, q - 1$.

Let K_1 and K_2 be two subcomplexes of L . We say that K_1 and K_2 are connected by a chain in L if there exists a vertex chain $v_1, \dots, v_q \in L$ such that $v_1 \in K_1$ and $v_q \in K_2$. The length of the shortest vertex chain connecting K_1 and K_2 is called the *chain distance* $d(K_1, K_2)$ between K_1 and K_2 .

DEFINITION 16.3.10. Let K be a subcomplex of an R -complex L . The number

$$\delta_K(L) = \max\{d(K, \bar{c}) \mid c \in (C(L) \setminus C(K)) \cup (FE(L) \setminus FE(K))\}$$

is called the *depth* of L with respect to K .

DEFINITION 16.3.11 (Depth of a disc diagram). Let M be a map. The number

$$\delta(M) = \delta_{\partial M}(M)$$

is called the *depth* of M .

The following result relates the depth and the radius of a diagram. Recall that the radius of a planar 2-complex L is defined as

$$\text{Rad}(L) = \max\{\rho(v, \partial L) \mid a \in V(L)\}$$

where ρ is the combinatorial distance in the 1-skeleton L^1 of L .

LEMMA 16.3.12. *Let \mathcal{D} be a van Kampen diagram over a finite symmetric presentation $\langle X; R \rangle$. Then*

$$\delta(\mathcal{D}) \leq \text{Rad}(\mathcal{D}) \leq \frac{1}{2}M(\mathcal{D})\delta(\mathcal{D}),$$

where $M(\mathcal{D}) = \max\{|r| \mid r \in R\}$.

Proof. To see that $\delta(\mathcal{D}) \leq \text{Rad}(\mathcal{D})$ it suffices to notice that if $p = e_1, \dots, e_k$ is a shortest edge-path in \mathcal{D} from a given vertex v to $\partial\mathcal{D}$ then the endpoints v_i, v_{i+1} of the edges e_i in p give a vertex chain $v = v_1, \dots, v_{k+1}$ in \mathcal{D} with $v_i, v_{i+1} \in \partial c_i$ where c_i is a cell containing the edge e_i . Therefore, $\delta(\mathcal{D}) \leq \text{Rad}(\mathcal{D})$. To prove the second inequality fix a vertex $v \in \mathcal{D}$ and consider a shortest vertex chain $v = v_1, \dots, v_{k+1}$ from v to $\partial\mathcal{D}$. Since every free edge is always on the boundary $\partial\mathcal{D}$ one may assume that $v_i, v_{i+1} \in \partial c_i$ where c_i is a cell in \mathcal{D} . It follows that for every pair v_i, v_{i+1} there is an edge-path in \mathcal{D} connecting these vertices (along the boundary of c_i) of length at most $\frac{1}{2}M(\mathcal{D})$. Hence the result. ■

As usual, one can introduce the *Depth filling function* $\Delta : \mathbb{N} \rightarrow \mathbb{N}$ of a finite presentation $\langle X; R \rangle$. To this end for a word $w \in F(X)$ such that $w = 1$ in $G = \langle X; R \rangle$ put

$$\delta(w) = \min\{\delta(\mathcal{M}) \mid \mathcal{M} \text{ a van Kampen diagram for } w\}$$

and define

$$\Delta(n) = \max\{\delta(w) \mid w =_G 1 \text{ and } |w| \leq n\}.$$

PROPOSITION 16.3.13. *A finitely generated group G is hyperbolic if and only if there is a constant $C > 0$ and a finite presentation $\langle X; R \rangle$ of G such that $\Delta(n) \leq C \log n$ for all n .*

Proof. The result follows from the inequalities in Lemma 16.3.12 and the following known result (see for example [32] Theorem 4.2.2): G is hyperbolic if and only if there is a constant $C > 0$ and a finite presentation $\langle X; R \rangle$ of G such that $\text{Rad}(n) \leq C \log n$ for all n . Here $\text{Rad}(n)$ is defined similarly to $\Delta(n)$ where the depth is replaced by the radius function of the diagrams. ■

Let $w = y_1 y_2 \dots y_k$ ($y_i \in X^{\pm 1}$) be a reduced word from $F(X)$. Define an X -digraph $\Gamma(w) = (V, E)$, where

- $V = \{y_1 \dots y_i \mid i = 0, \dots, k\};$
- $E = \{y_1 \dots y_i \xrightarrow{y_{i+1}} y_1 \dots y_i y_{i+1} \mid i = 0, \dots, k-1\}.$

The graph $\Gamma(w)$ is a straight line segment labeled with a word w . Clearly, there is a graph morphism $\Gamma(w) \rightarrow \Gamma(G, X)$.

PROPOSITION 16.3.14. *Let D be a diagram with a boundary label w , $m = \delta(D)$, and $\varphi : \Gamma(w) \rightarrow \partial D$ an X -digraph morphism. Then there exists a morphism of 2-complexes $\psi : D \rightarrow \mathcal{C}^{(m)}(\Gamma(w))$ such that the following diagram commutes:*

$$\begin{array}{ccc} \Gamma(w) & \xrightarrow{\varphi} & D \\ & \searrow \varphi c & \downarrow \psi \\ & & \mathcal{C}^{(m)}(\Gamma(w)) \end{array}$$

Proof. To prove the assertion of the proposition we first cut a diagram D into a diagram T of a certain type. The diagram T is a forest of cells attached at a line segment graph labeled with w (see Figure 16.2). The height of the forest (distance from the line segment to cells) is at most m . We denote the corresponding sewing morphism by $\theta : T \rightarrow D$ (read more about cuts and sewing morphisms in Section 16.6.2).

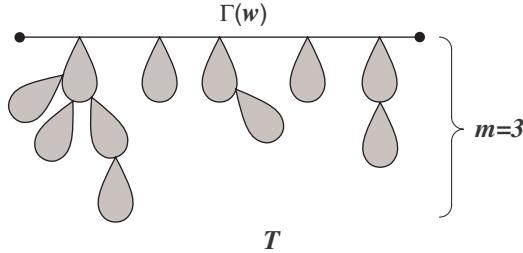
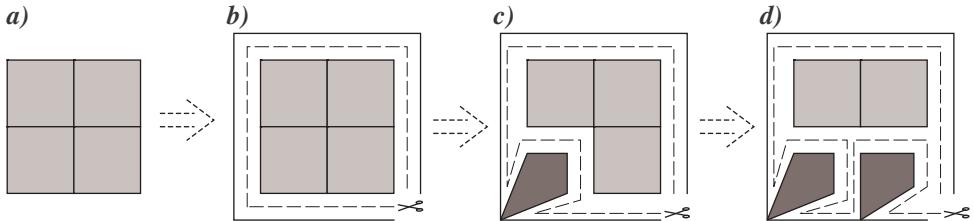


FIGURE 16.2. A “forest attached to a line segment” diagram.

We construct the loop along which we cut D in a sequence of steps. Denote by π_0 the boundary loop ∂D starting at the initial vertex of $\varphi(\Gamma(w))$ (from which w is read). The loop π_0 cuts off a line segment labeled with w (see Figure 16.3.b).

FIGURE 16.3. First steps of constructing T .

Assume that π_i is the last constructed loop and T_i is a diagram of a specified type cut off D by π_i . If T_i includes all cells of D then T_i is a required diagram T . Otherwise choose a cell c which does not belong to T_i with the smallest value $d = d(\partial D, c)$. If $d = 1$ then c touches the boundary ∂D at some vertex v in D . In this case pull π_i over c as shown in Figure 16.3. If $d > 1$ then c touches at some vertex v some cell c' such that $d' = d(\partial D, c') = d - 1$. The cell c' already belongs to T_i . Pull π_i over c at v . Clearly, the obtained loop π_{i+1} cuts off a diagram T_{i+1} which includes one additional cell. We continue this process until we have no cells to add to T_i .

Since the result of Stallings’ procedure does not depend on a sequence of folds it follows that the following diagram commutes:

$$\begin{array}{ccc}
 T & \xrightarrow{\theta} & D \\
 & \searrow \varphi_S & \downarrow \varphi_S \\
 & & S(T)
 \end{array}
 \tag{34}$$

Now it follows from the definition of $\mathcal{C}_1^{(m)}$ and T that there exists a morphism $\theta' : T \rightarrow \mathcal{C}_1^{(m)}(\Gamma(w))$ such that the following diagram commutes:

$$(35) \quad \begin{array}{ccc} \Gamma(w) & \hookrightarrow & T \\ & \searrow \varphi_{\mathcal{C}_1} & \downarrow \theta' \\ & & \mathcal{C}_1^{(m)}(\Gamma(w)) \end{array}$$

Using the diagrams (34) and (35) and the fact that operator S is a functor, there exists a morphism $\tau : S(T) \rightarrow S(\mathcal{C}_1^{(m)}(\Gamma(w)))$ such that the following diagram commutes:

$$\begin{array}{ccccc} \Gamma(w) & \hookrightarrow & T & \xrightarrow{\theta} & D \\ & \searrow \varphi_{\mathcal{C}_1} & \downarrow \theta' & \swarrow \varphi_S & \downarrow \varphi_S \\ & & \mathcal{C}_1^{(m)}(\Gamma(w)) & S(T) & \\ & & \searrow \varphi_S & \downarrow \tau & \\ & & & S(\mathcal{C}_1^{(m)}(\Gamma(w))) & \end{array}$$

By Lemma 16.3.5 $S(\mathcal{C}_1^{(m)}(\Gamma(w))) = \mathcal{C}^{(m)}(\Gamma(w))$ which finishes the proof. ■

16.4. New algorithms for the word search problem in groups

The word problem, the conjugacy problem, and the membership problem are classical algorithmic problems in groups. We refer to surveys [1], [199], and [200] on algorithmic problems in groups.

16.4.1. Search problems in groups. Let G be a fixed group given by a symmetrized finite presentation $G = \langle X; R \rangle$ and $M(X) = (X^{\pm 1})^*$ a free monoid over the alphabet $X^{\pm 1}$. Sometimes, slightly abusing notation, we identify words in $M(X)$ with their canonical images in the free group $F(X)$.

An algorithmic problem P over G can be described as a subset $D = D_P$ of a Cartesian power $M(X)^k$ of $M(X)$. The problem is *decidable* if there exists a *decision algorithm* $A = A_P$ which on a given input $w \in M(X)^k$ halts and outputs “Yes” if $w \in D_P$, otherwise it outputs “No”. It is convenient to view P as consisting of two parts: the “Yes” (or *positive*) part requires a partial algorithm A_{Yes} which on an input $w \in D$ halts and outputs “Yes”, and works forever on inputs from $M(X)^k - D$; the “No” (negative) part asks for a partial algorithm for the set $M(X)^k - D$. Recently, it has been shown that for a wide variety of finitely presented groups the “No” part is very easy on *average*, as well as *generically* (see [146], [147]). On the other hand, in many applications it is required to find a decision algorithm A_{Yes} for the “Yes” part of P . Furthermore, very often one has to find a decision algorithm A_{Yes} which on an input $w \in D$ provides a “reasonable proof” that w is, indeed, in D . This leads to the so-called *search* (or *witness*) variations of the algorithmic problems (see [146], [147], and [31] for a more detailed discussion of the search problems in groups):

Word Search Problem (WSP) for $G = \langle X; R \rangle$: For a given $w \in M(X)$ satisfying $w =_G 1$ represent w as a product of conjugates of relators from R .

Conjugacy Search Problem (CSP) for G : For a given pair $(u, v) \in M(X) \times M(X)$ satisfying $u \sim_G v$, find a conjugator for u and v .

Membership Search Problem (MSP) for G : For a given word $w \in M(X)$ and a finitely generated subgroup H of G (given by a finite set of generators) such that $w \in H$, represent w as a product of the generators of H .

This new aspect of the search decision problems, which requires a “proof” or a “witness”, of the correct decision, needs a more detailed explanation. Let $D \subseteq M(X)^k$ be a search decision problem, \mathcal{A} a decision algorithm for D , $w \in D$ a particular instance of the “Yes” part of the problem, and p_w a “proof” provided by \mathcal{A} that w , indeed, belongs to D . The time complexity function $T_{\mathcal{A}}(x)$ of \mathcal{A} typically takes into account the time required for \mathcal{A} to check whether or not $w \in D$, as well as the time needed for \mathcal{A} to produce p_w on the input w . Hence the time complexity of \mathcal{A} depends on the complexity of the routine to produce p_w , in particular, on the way one represents these p_w as objects (words, graphs, sequence of formal derivations, programs, etc.). It is not clear what kind of representations are the most convenient in computations, this, perhaps, depends on a particular problem and the decision algorithm. But there is another important problem here. Namely, when given a proof p_w it might take a considerable amount of time to confirm, using p_w , that w belongs to D , so obtaining a proof and verification of the decision based on this proof are different processes. Should one add this verification time to the time complexity of the algorithm \mathcal{A} or treat it as a separate issue, is not altogether clear. It is surprising how little was done on this topic in computational algebra. It may happen (see the search membership problem for free groups below) that the time function $T_V(w, p_w)$ for verification process on the inputs w, p_w is exponentially greater than the time function $T_{\mathcal{A}}(w)$ of the process of constructing the proof p_w . In what follows we treat the time complexities of the decision algorithm \mathcal{A} and that of the verification process as different issues.

16.4.2. The word search problem in groups. Algorithm \mathcal{A} . In this section we introduce a new algorithm for the word search problem (**WSP**) in groups and study its worst-case time complexity. Given a finite symmetrized presentation $G = \langle X; R \rangle$ and a word $w \in gp_F(R)$ the algorithm \mathcal{A} produces a proof p_w which is a finite folded X -digraph Γ such that Γ approximates the Cayley graph $\Gamma(G, X)$ and which accepts w (there is a loop with label w at the base point in Γ). In the worst-case scenario the size of the graph $p_w = \Gamma$ is exponential in the length of w , but given Γ and w the verification procedure is linear in the length of w (one needs only to read w in Γ viewed as a deterministic finite automaton). In this case, the verification time function $T_V(w, p_w)$ is negligible compared to the time function $T_{\mathcal{A}}$.

We would like to emphasize that Algorithm \mathcal{A} is the most efficient general technique for WSP so far. In fact, there are just a few techniques that work for any finite presentation, e.g., Todd-Coxeter algorithm, total enumeration of $gp_F(R)$, Knuth-Bendix procedure, and their numerous modifications. Algorithm \mathcal{A} is itself a modification of a standard Todd-Coxeter procedure. At the end of the chapter (in Section 16.9) we make a short comparison of \mathcal{A} with Todd-Coxeter and enumeration of $gp_F(R)$.

We start with a brief description of the algorithm \mathcal{A} . Suppose we are given a word $w \in gp_F(R)$. To provide a proof that w belongs to $gp_F(R)$ it suffices to find an approximation Γ of $\Gamma(G, X)$ in which there is a closed path with the label w . Indeed, in this event there exists a closed path with the label w in $\Gamma(G, X)$ so $w \in gp_F(R)$. Now, given a word $w \in F(X)$ the algorithm \mathcal{A} begins to construct some particular approximations Γ of $\Gamma(G, X)$; it stops in finitely many steps if $w \in gp_F(R)$ and works forever otherwise. If \mathcal{A} stops on an input w then the output of \mathcal{A} is an approximation Γ of $\Gamma(G, X)$ in which there exists a loop with label w at the base-point of Γ . The principle idea behind this algorithm is that we do not enumerate all possible approximations Γ of $\Gamma(G, X)$ (as happens in the Todd-Coxeter algorithms), but rather we construct only those approximations which contain a path at the based-point (perhaps, not closed) with the label w .

Recall that $\Gamma(w)$ is an X -digraph which is a line segment labeled with w (see Section 16.3.3). The starting vertex 1 of the segment is called the *base-point* of the graph $\Gamma(w)$. The canonical image of 1 in $\mathcal{C}^{(m)}(\Gamma(w))$ is the base-point of $\mathcal{C}^{(m)}(\Gamma(w))$, which we denote again by 1.

We start with the following algorithm.

ALGORITHM 16.4.1 (Decision Algorithm \mathcal{A} for **WSP** in groups).

INPUT: A finite symmetrized presentation $\langle X; R \rangle$, and $w \in F(X)$.

OUTPUT: YES if $w \in gp_F(R)$ and a finite approximation Γ of $\Gamma(G, X)$ which accepts w .

COMPUTATIONS:

- Consequently compute $\mathcal{C}^{(i)}(\Gamma(w))$ until the endpoints of the image of $\Gamma(w)$ become equal in $\mathcal{C}^{(i)}(\Gamma(w))$.
- Return YES and $\mathcal{C}^{(i)}(\Gamma(w))$.

DEFINITION 16.4.2. For a word $w \in gp_F(R)$ define a number

$$\delta(w) = \min\{\delta(D) \mid D \text{ is a diagram over } \langle X; R \rangle \text{ with boundary label } w\}.$$

The number $\delta(w)$ is called the *depth* of the word w in G .

Observe that by Lemma 16.3.7, $\delta(w)$ is defined for any $w \in gp_F(R)$.

THEOREM 16.4.3. *The Decision Algorithm \mathcal{A} needs at most $m = \delta(w)$ iterations to stop on an input $w \in gp_F(R)$. The total number of steps required by the algorithm to stop on an input $w \in gp_F(R)$ is bounded from above by*

$$O(m|w| \cdot L(R)^m \log(|w|L(R))).$$

Proof. Let $D(w)$ be a van Kampen diagram such that the label of the boundary ∂D (at some vertex v on ∂D) is w and $\delta(w) = \delta(D(w))$. By Proposition 16.3.14 there exists an embedding of $D(w)$ into $\Gamma = \mathcal{C}^{(m)}(\Gamma(w))$, where $m = \delta(D(w))$, such that the image of the vertex v is the based-point of Γ . This proves the first part of the theorem. The second part follows from Lemma 16.3.4. This completes the proof. ■

Notice, that it is not easy to extract $D(w)$ from $\mathcal{C}^{(m)}(\Gamma(w))$ even when a morphism $D(w) \rightarrow \mathcal{C}^{(m)}(\Gamma(w))$ does exist. In the next section we describe an algorithm to do just that.

16.4.3. The word search problem in groups. **Algorithm \mathcal{B} .** In this section we describe a new search decision algorithm \mathcal{B} for **WSP** in groups and study its worst-case time complexity. Given a finite symmetrized presentation $G = \langle X; R \rangle$ and a word $w \in gp_F(R)$ the algorithm \mathcal{B} produces a proof $p_{\mathcal{B}}(w)$ which is a finite sequence of derivations of a certain type. The sequence $p_{\mathcal{B}}(w)$ allows one to rewrite w as a product of conjugates of relators from R ; this is the verification process for \mathcal{B} on the inputs $w, p_{\mathcal{B}}(w)$. Thus, the algorithm \mathcal{B} together with the verification procedure for a given $w \in gp_F(R)$ output a decomposition of w as a product of conjugates of relators from R . This is a much stronger result than the one provided by the algorithm \mathcal{A} . Notice that the size of the proof $p_{\mathcal{B}}(w)$ is comparable to the size of $p_{\mathcal{A}}(w)$, but the verification process is much more time consuming than in \mathcal{A} . The worst-case time complexity of \mathcal{B} is still exponential in the length of w . To the best of our knowledge \mathcal{B} is the first general algorithm for **WSP** in groups with a single exponential upper bound on the worst-case complexity.

A brief description of the algorithm \mathcal{B} is the following. Given $\langle X; R \rangle$ and $w \in gp_F(R)$ one starts the algorithm \mathcal{A} on the input w . The algorithm \mathcal{A} returns an X -digraph Γ which accepts w . The graph Γ gives Stallings' folding (with respect to X) of some finitely generated subgroup H of $gp_F(R)$ which contains w . Afterward, one uses the standard algorithm for membership search problem for finitely generated subgroups of free groups to find a presentation of w as a product of conjugates of relators from R . This solves **WSP** in G for w in the classical formulation.

Now we give a formal description of the algorithm. For an X -digraph Γ with the base-point 1 define the *radius* $r(\Gamma)$ of Γ to be the maximum of the distances $d(1, u)$, $u \in \Gamma$, where d is the standard graph metric on Γ .

LEMMA 16.4.4. *Let $\langle X; R \rangle$ be a finite symmetrized presentation, Γ a finite X -digraph with a base-point 1, H the subgroup in $F(X)$ accepted by $\text{Core}(\Gamma)$. Then the X -digraph $\mathcal{C}(\Gamma)$ (with the base-point induced from Γ) accepts a subgroup of $F(X)$ generated by $H \cup A$, where A is a finite set of conjugates of elements from R satisfying the following conditions:*

- (1) $|A| \leq |V(\Gamma)| \cdot |R|$.
- (2) *For any $a \in A$, $|a| \leq 2r(\Gamma) + M(R)$.*
- (3) *One can find such a generating set A effectively in time*

$$O(|V(\Gamma)| \cdot |R| \cdot (2r(\Gamma) + M(R))).$$

Proof. The graph $\mathcal{C}(\Gamma)$ is obtained from Γ by adding to each vertex $v \in \Gamma$ a loop with the label r for each $r \in R$, which follow by several consecutive foldings. Let c be a loop with the label $r \in R$ added to Γ at a vertex $v \in \Gamma$. Let p_v be the label of a shortest path from the base-point 1 to v in Γ . Adding c to Γ results in the same graph as if adding a loop labeled by $p_v r p_v^{-1}$ to the base-point of Γ and folding the path with the label p_v into Γ . Since the resulting graph does not depend on a particular sequence of foldings (because $\mathcal{C}_1(\Gamma)$ is a core graph) one can readily see that $\mathcal{C}(\Gamma)$ accepts a subgroup of $F(X)$ generated by H together with a finite set $A = \{p_v r p_v^{-1} \mid v \in V(\Gamma), r \in R\}$. Obviously, $|A| \leq |V(\Gamma)| \cdot |R|$ which proves (1).

Since $|p_v| \leq r(\Gamma)$ and $|r| \leq M(R)$ then for every $a = p_v r p_v^{-1} \in A$ we have $|a| \leq |p_v| + |r| + |p_v^{-1}| \leq 2r(\Gamma) + M(R)$. Hence (2) and (3) follow. ■

PROPOSITION 16.4.5. *Let $w \in F(X)$. Then the graph $\mathcal{C}^{(m)}(\Gamma(w))$ is the subgroup graph of a subgroup of $F(X)$ generated by a finite set A of elements from R and their conjugates. Moreover,*

- (1) $|A| \leq (|w| + 1)|R|^m$.
- (2) If $a \in A$ then $|a| \leq 2|w| + mM(R)$.
- (3) One can find a generating set A effectively in time

$$O(|w| \cdot |R|^m \cdot (2|w| + mM(R))).$$

Proof. (1) and (2) clearly follow from Lemma 16.4.4 and observation that $|V(\Gamma(w))| = |w| + 1 = O(|w|)$. To show (3) notice that adding a loop of length l to an X -digraph increases the radius of the graph at most by $\lfloor M(R)/2 \rfloor$. Now the result follows from this observation and the fact that the radius of the initial graph $\Gamma(w)$ is $|w|$. ■

Assume that for some $m \in \mathbb{N}$ the graph $\Gamma' = \mathcal{C}^{(m)}(\Gamma(w))$ accepts w . Let H be the subgroup in $F(X)$ accepted by Γ' and A a set of conjugates of elements from R from Proposition 16.4.5 that generates the subgroup H . Clearly, $w \in H$ and to present w as a product of conjugates of relators from R it suffices to solve the membership search problem (**MSP**) for the subgroup $H = \langle A \rangle$ on the given input w . The standard way of solving **MSP** in free groups involves Nielsen minimization method. It gives exponential time estimates on the worst-case complexity. Let's have a look where the exponential time estimates come from in the Nielsen argument. It takes quadratic time for a given tuple of generators $h = (h_1, \dots, h_k)$ of a subgroup H to find a sequence η of Nielsen moves η_1, \dots, η_s which reduces the set h to a Nielsen basis $f = (f_1, \dots, f_t)$ of H . It takes also at most quadratic time to express w as a word in new generators f . Now, to express w in the old generators h_i one needs to rewrite the new generators f_i as words in h_i , but the length of the resulting words can grow exponentially in terms of $|h|$. Notice, however, that the sequence of Nielsen moves η completely describes the formal expressions of f_i 's in terms of h , and the length of η is linear in $|h|$. This leads to the idea to use the sequence η in producing the proof $p_B(w)$ rather than the whole expression of w as a word in h .

To describe this alternative way of producing $p_B(w)$ we need to introduce *derivation rules*. Let

- H be a subgroup of $F(X)$ generated by a finite set $A = \{a_1, \dots, a_n\} \subset F(X)$,
- $\tilde{A} = \{\alpha_1, \dots, \alpha_n\}$ be a set of formal names for words from A together with a homomorphism $\varphi_A : F(\tilde{A}) \rightarrow \langle A \rangle$ defined by $\varphi_A(\alpha_i) = a_i$,
- $\Gamma(A)$ be an X -digraph which is the wedge of n loops labeled with words $a_i(X)$.
- $Q_0 = \{q_1, \dots, q_s\}$ be a set,
- $Q = Q_0 \cup \tilde{A} \cup E(S(\Gamma(A)))$.

A *derivation rule* is a pair $(\alpha, \beta) \in Q \times Q^*$ of one of the following types:

- (e, q_i) , where $e \in E(S(\Gamma(A)))$ and $q_i \in Q_0$;
- $(q_i, q_j q_k)$, where $q_i, q_j, q_k \in Q_0$ and $i > \max\{j, k\}$;
- (q_i, α) , where $q_i \in Q_0$ and $\alpha \in \tilde{A}$.

DEFINITION 16.4.6. A *derivation system* W is a set of derivation rules satisfying the following properties:

- (D1) for each edge $e \in E(S(\Gamma(A)))$ there exists one rule with the left side e ;
- (D2) if $q_i \in Q_0$ is involved in the right side of some rule in W then there exists one rule with the left side q_i .

As usual a derivation is a sequence of applications of the derivation rules to a word from Q^* . Clearly any derivation sequence terminates on any word u from Q^* . The resulting word u^* does not depend on a derivation process. Given a derivation system W one can define a map α on the set of edges from $S(\Gamma(A))$ by $e \rightarrow e^*$. The map α naturally extends to the set of all paths in $S(\Gamma(A))$:

$$e_1 \dots e_k \xrightarrow{\alpha} e_1^* \dots e_k^*.$$

Now if $a \in \langle A \rangle$ then there exists a unique path p_a in $S(\Gamma(A))$ with the label a starting at the base-point 1 in $S(\Gamma(A))$. Define $\alpha : A \rightarrow F(\tilde{A})$ by $a \xrightarrow{\alpha} p_a^*$.

A derivation system W is said to be *compatible* with $\langle A \rangle$ if for every $w \in \langle A \rangle$ $\alpha(w) \in F(\tilde{A})$ correctly represents w in $F(X)$, i.e., $\varphi_A(\alpha(w)) = w$. Clearly, the graph $S(\Gamma(A))$ and a derivation system W compatible with $\langle A \rangle$ give a straightforward solution to **MSP** for $\langle A \rangle$.

PROPOSITION 16.4.7. *Let H be a subgroup of $F(X)$ generated by a finite set $A = \{a_1, \dots, a_n\} \subset F(X)$. Then one can effectively find a finite derivation system compatible with H in at most quadratic time $O(L(A)^2)$.*

Proof. Applying Stallings' folding process to the graph $\Gamma(A)$ we obtain a sequence of intermediate graphs:

$$\Gamma(A) = \Gamma_0 \rightarrow \Gamma_1 \rightarrow \dots \rightarrow \Gamma_m = S(\Gamma(A)),$$

where $m \leq L(A)$. First, define a system of derivation rules for Γ_0 . Choose an edge e_{a_i} on a loop for a_i for every $a_i \in A$ and put

$$W_0 = \{(e_{a_i}, \alpha_i), (e_{a_i}^{-1}, \alpha_i^{-1}) \mid i = 1, \dots, n\}.$$

Clearly, W_0 is compatible with $\langle A \rangle$. Assume that we have a compatible system W_i for the graph Γ_i . It is straightforward to construct a compatible system of rules for Γ_{i+1} satisfying $|W_{i+1}| - |W_i| \leq |E(\Gamma_0)| \leq L(A)$. Hence the result. ■

Propositions 16.4.5 and 16.4.7 give the following result.

PROPOSITION 16.4.8. *Let $G = \langle X; R \rangle$, $w \in F(X)$, and $\mathcal{C}^{(m)}(\Gamma(w))$ a finite approximation of $\Gamma(G, X)$ which accepts w . Then one can effectively find a set of free generators A for $\mathcal{C}^{(m)}(\Gamma(w))$ with a derivation system compatible with $\langle A \rangle$ in at most $O(|w|^2|R|^{2m}(2|w| + mM(R))^2)$ steps.*

Proof. Consider a set A of generators as in proof of Lemma 16.4.4. Computation of such a set requires at most $O(M(R)(|w| + 1)|R|^m)$ steps. Clearly, $L(A) \leq |A| \max_{a \in A} \{|a|\}$ and hence, using estimates in Proposition 16.4.5, we get $L(A) \leq (|w| + 1)|R|^m(2|w| + mM)$. By Proposition 16.4.7 the complexity of computing a derivation system for A is $O(|w|^2|R|^{2m}(2|w| + mM(R))^2)$. ■

ALGORITHM 16.4.9 (Decision Algorithm \mathcal{B} for **WSP** in groups).

INPUT. A finite symmetrized presentation $\langle X; R \rangle$ and a word $w \in gp_F(R)$.

OUTPUT. A finite approximation Γ of $\Gamma(G, X)$ which accepts w , the set A of free generators of the subgroup H and a derivation system W compatible with A (as described in Proposition 16.4.5).

COMPUTATIONS.

- 1) Compute the approximation $\Gamma = \mathcal{C}^{(m)}(\Gamma(w))$ which accepts w using the algorithm \mathcal{A} .

- 2) Compute a free set of generators A of the subgroup H accepted by Γ as described in Proposition 16.4.5.
- 3) Compute the system of derivations W compatible with A as in Proposition 16.4.7.

Combining Theorem 16.4.3 and Proposition 16.4.8 we obtain the following result.

THEOREM 16.4.10. *Given a finite symmetrized presentation $\langle X; R \rangle$ and an element $w \in gp_F(R)$ the algorithm \mathcal{B} outputs a finite set A of conjugates of elements from R , the subgroup graph of $\langle A \rangle$ in $F(X)$ which accepts w , and a finite derivation system W compatible with A . The worst-case complexity of this algorithm is bounded by*

$$O(|w|^2|R|^{2\delta(w)}(2|w| + \delta(w)M(R))^2).$$

16.5. Random van Kampen diagrams

In this section we describe a class of stochastic procedures, so-called *iterative random generators* which generate random van Kampen diagrams over a given finite presentation $\langle X; R \rangle$. Roughly speaking, a random generator RG starts with a given diagram D_0 and then randomly extends it according to some basic pattern (*basic random extension*). Usually a given random generator depends on a set of distributions that allows one to obtain random diagrams with various properties. One can view such iterative random generators as random walks on corresponding *transition graphs*. This allows one to introduce a measure on the set of the corresponding trajectories and then induce this measure on the diagrams produced by the generators.

16.5.1. Basic random extensions and simple random walks. In this section we define random walks on diagrams and probability spaces on sequences of diagrams. Later it will be used to define asymptotic density on diagrams.

Let $\mathcal{K} = \{D_i \mid i \in \mathbb{N}\}$ be a countable (enumerable) collection of diagrams. Here we assume that diagrams from \mathcal{K} are van Kampen diagrams over some fixed presentation $\langle X; R \rangle$ equipped, perhaps, with some extra predicates. Denote by $B : \mathcal{K} \rightarrow \mathcal{K}$ a stochastic map that with probability $p_{i,j}$ maps D_i into D_j . Sometimes we write $B(D_i) = D_j$ if $p_{i,j} > 0$. The map B can be viewed as a random walk on \mathcal{K} defined by the infinite stochastic matrix $(p_{i,j})$. We say that B is a *basic extension* if for any $D_i, D_j \in \mathcal{K}$ such that $B(D_i) = D_j$ there exists a diagram morphism $D_i \rightarrow D_j$. Given a basic extension B we define the *transition graph* $T_B = (V, E)$ of B which is a directed weighted graph defined as follows:

- 1) $V = \mathcal{K}$;
- 2) $E = \{(D_i, D_j) \mid p_{ij} > 0\}$;
- 3) each edge $e = (D_i, D_j) \in E$ has an associated number $p(e) = p_{ij}$.

For $D \in \mathcal{K}$ we denote by $\Phi = \Phi_B(D)$ the set of all diagrams C in \mathcal{K} such that there exists a path in T_B from D to C . A basic extension B is called \mathcal{K} -complete if there exists $D \in \mathcal{K}$ such that $\Phi_B(D) = \mathcal{K}$. More generally, if $\varphi : \mathcal{K} \rightarrow \mathcal{L}$ is a mapping from \mathcal{K} onto a collection of diagrams \mathcal{L} then we say that B is \mathcal{L} -complete relative to φ if $\varphi(\Phi) = \mathcal{L}$.

Recall that the neighborhood $N_{\mathcal{M}}(v)$ of a vertex v in a diagram \mathcal{M} is defined as the submap generated by all edges and faces in \mathcal{M} incident to v . Let $D \in \mathcal{K}$ and v

be a vertex in D . We say that B is *locally stable* at the vertex v if the neighborhood of v eventually stabilizes, i.e., for any infinite path

$$D = C_1 \rightarrow C_2 \rightarrow \dots$$

in the graph T_B there exists $j_0 \in \mathbb{N}$ such that $N_{C_j}(v) = N_{C_{j_0}}(v)$ for every $j \geq j_0$. A random generator B is called *locally stable* if it is stable at every vertex v of every diagram $D \in \mathcal{K}$.

Given a basic extension B and a diagram $D_0 \in \mathcal{K}$ define a new transition graph (which depends on B and D_0) $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$, where

$$V(\mathcal{T}) = \{p \mid p \text{ is a finite path in } T_B \text{ starting at } D_0\}$$

and

$$E(\mathcal{T}) = \{(\theta, \theta e) \mid \theta, \theta e \in V(\mathcal{T}), e \in E(T_B)\}.$$

Clearly, \mathcal{T} is a tree. We will refer to \mathcal{T} as the *transition tree* of B (with the empty path ε in the root). For each edge $d = (\theta, \theta e) \in E(\mathcal{T})$ we assign probability $p'(d) = p(e)$.

By $\mathcal{W} = \mathcal{W}_{\mathcal{T}}$ we denote a random walk on the tree \mathcal{T} defined by the transition probabilities p' (we assume here that \mathcal{W} starts with probability 1 at the root ε). As usual one can view the random walk \mathcal{W} as a sequence of random variables Z_n , $n \in \mathbb{N}$, on a suitable probability space $(\Lambda, \mathcal{F}, P)$. To explain this we need a few definitions. An infinite path λ in the directed graph \mathcal{T} which starts at the root ε is called a *trajectory*. For a trajectory λ by λ_i we denote the vertex on λ at distance i from the root ε (the i -th component of λ). Now, Λ is the set of all trajectories in \mathcal{T} and $Z_n : \Lambda \rightarrow V(\mathcal{T})$ is a random variable such that for $\lambda \in \Lambda$ one has $Z_i(\lambda) = \lambda_i$. A *cone* of $\theta \in V(\mathcal{T})$ is the set of all trajectories passing through θ :

$$Cone(\theta) = \{\lambda \in \Lambda \mid \lambda \text{ is passing through } \theta\}.$$

The σ -algebra \mathcal{F} is generated by all cones $Cone(\theta)$, where $\theta \in V(\mathcal{T})$. For each $\theta = e_1 \dots e_k \in V(\mathcal{T})$ the real number $P(Cone(\theta))$ is defined as the probability to hit the vertex $\theta \in \mathcal{T}$ by the random walk \mathcal{W} , i.e.,

$$P(Cone(\theta)) = \prod_{i=1}^k p(e_i).$$

By the Kolmogorov's extension theorem the function P extends onto the σ -algebra \mathcal{F} in such a way that $(\Lambda, \mathcal{F}, P)$ is a probability space, so P is a probability measure on Λ .

16.5.2. Probability and asymptotic measure on diagrams. In this section we define discrete probability measures and asymptotic densities on the sets $V(\mathcal{T})$, \mathcal{K} , and \mathcal{L} for a given map $\varphi : \mathcal{K} \rightarrow \mathcal{L}$.

16.5.2.1. Probability on $V(\mathcal{T})$. Let (Λ, P) be the probability space defined in the previous section and $Q : \Lambda \rightarrow \mathbb{N}$ a random variable on Λ . We view the function Q as a *termination condition* for the random walk \mathcal{W} which shows where the walk stops going along a path λ . Define a function $T_Q : \Lambda \rightarrow V(\mathcal{T})$ by

$$T_Q(\lambda) = \lambda_{Q(\lambda)}$$

for $\lambda \in \Lambda$.

LEMMA 16.5.1. *For any $\lambda \in \Lambda$ the set $T_Q^{-1}(\lambda_{Q(\lambda)})$ is measurable in (Λ, P) .*

Proof. Follows from the equality

$$T_Q^{-1}(\lambda_{Q(\lambda)}) = Q^{-1}(Q(\lambda)) \cap \text{Cone}(\lambda_{Q(\lambda)})$$

and the assumption that Q is a random variable. \blacksquare

Denote by V_Q the set of all stop-vertices of \mathcal{W} in \mathcal{T} relative to the termination condition Q , so

$$V_Q = T_Q(\Lambda) \subset V(\mathcal{T}),$$

and define a function $P_Q : V_Q \rightarrow \mathbb{R}$ by

$$P_Q(\theta) = P(T_Q^{-1}(\theta))$$

for $\theta \in V_Q$.

PROPOSITION 16.5.2. *The function P_Q is a discrete probability measure on V_Q .*

Proof. By Lemma 16.5.1 $P_Q(\theta)$ is defined and non-negative for every $\theta \in V_Q$. Clearly, if $\theta_1 \neq \theta_2$ then $T_Q^{-1}(\theta_1) \cap T_Q^{-1}(\theta_2) = \emptyset$, so

$$\bigcup_{\theta \in V_Q} T_Q^{-1}(\theta) = \Lambda$$

is a partition of Λ . Hence

$$P_Q(V_Q) = \sum_{\theta \in V_Q} P_Q(\theta) = 1,$$

as required. \blacksquare

Let $Q_i : \Lambda \rightarrow \mathbb{N}$, $i \in \mathbb{N}$, be a sequence of random variables (termination conditions) on Λ such that

$$(36) \quad V(\mathcal{T}) = \bigcup_{i \in \mathbb{N}} V_{Q_i}, \quad V_{Q_i} \cap V_{Q_j} = \emptyset \quad (i \neq j).$$

In this event we say that the sequence $\mathcal{Q} = \{Q_i\}_{i \in \mathbb{N}}$ of termination conditions for \mathcal{W} is *complete*. The complete sequence of termination conditions \mathcal{Q} allows one to define an *asymptotic density* ρ on $V(\mathcal{T})$ with respect to \mathcal{Q} . Namely, if $S \subseteq V(\mathcal{T})$ then the asymptotic density $\rho(S)$ of S in $V(\mathcal{T})$ relative to B , D_0 , and \mathcal{Q} is equal to the following limit (if it exists)

$$\rho(S) = \lim_{i \rightarrow \infty} P_{Q_i}(S \cap V_{Q_i}).$$

Let $\mu : \mathbb{N} \rightarrow \mathbb{R}$ be a fixed probability distribution on \mathbb{N} . Define a probability measure

$$P_{V(\mathcal{T})} : V(\mathcal{T}) \rightarrow \mathbb{R}$$

which depends on \mathcal{Q} and μ as follows. For $\theta \in V(\mathcal{T})$ such that $\theta \in V_{Q_i}$ for some $i \in \mathbb{N}$ put

$$(37) \quad P_{V(\mathcal{T})}(\theta) = \mu(i)P_{Q_i}(\theta).$$

PROPOSITION 16.5.3. *The function $P_{V(\mathcal{T})}$ is a discrete probability measure on the set $V(\mathcal{T})$.*

Proof. Clearly, the value $P_{V(\mathcal{T})}(\theta)$ is defined for every $\theta \in V(\mathcal{T})$ and is non-negative. Therefore, it suffices to show that $\sum_{\theta \in V(\mathcal{T})} P_{V(\mathcal{T})}(\theta) = 1$. The latter comes from the following equalities:

$$\sum_{\theta \in V(\mathcal{T})} P_{V(\mathcal{T})}(\theta) = \sum_{i \in \mathbb{N}} \sum_{\theta \in V_{Q_i}} P_{V(\mathcal{T})}(\theta) = \sum_{i \in \mathbb{N}} \sum_{\theta \in V_{Q_i}} \mu(i) P_{Q_i}(\theta) = \sum_{i \in \mathbb{N}} \mu(i) = 1.$$

■

16.5.2.2. *Probability on \mathcal{K} .* Next we define a probability measure $P_{\mathcal{K}}$ on diagrams \mathcal{K} . Let $\theta \in V(\mathcal{T})$. By definition $\theta = e_1 \dots e_k$ is a path in the transition graph $T = T_B$ with the origin D_0 and the terminus $D(\theta)$. For $D \in \mathcal{K}$ we define a function

$$P_{\mathcal{K}}(D) = \sum_{\theta \in V(\mathcal{T}), D(\theta)=D} P_{V(\mathcal{T})}(\theta)$$

(here we assume that $P_{\mathcal{K}}(D) = 0$ if there is no $\theta \in V(\mathcal{T})$ such that $D(\theta) = D$). The next lemma is obvious.

LEMMA 16.5.4. *The function $P_{\mathcal{K}}$ is a discrete probability measure on \mathcal{K} .*

Finally, define sets

$$\mathcal{K}_i = D(V_{Q_i}) = \{D(\theta) \mid \theta \in V_{Q_i}\}$$

with functions $P_{\mathcal{K}_i} : \mathcal{K}_i \rightarrow \mathbb{R}$ such for $D \in \mathcal{K}_i$:

$$P_{\mathcal{K}_i}(D) = \sum_{\theta \in V_{Q_i} \text{ and } D(\theta)=D} P_{Q_i}(\theta).$$

PROPOSITION 16.5.5. *The function $P_{\mathcal{K}_i}$ is a discrete probability measure on \mathcal{K}_i .*

Notice that if the sets \mathcal{K}_i form a partition of \mathcal{K} then one can define an *asymptotic density* of diagrams from \mathcal{K} as follows. If $S \subseteq \mathcal{K}$ then

$$\rho(S) = \lim_{i \rightarrow \infty} P_{\mathcal{K}_i}(\mathcal{K}_i \cap S)$$

(if it exists) is an asymptotic density of S in \mathcal{K} .

16.5.2.3. *Probability on \mathcal{L} .* Let $\varphi : \mathcal{K} \rightarrow \mathcal{L}$ be a mapping from \mathcal{K} into a collection of diagrams \mathcal{L} . We induce a probability $P_{\mathcal{L}}$ on \mathcal{L} from \mathcal{K} through φ : for $D \in \mathcal{L}$

$$P_{\mathcal{L}}(D) = \sum_{\theta \in V(\mathcal{T}) \text{ and } \varphi(D(\theta))=D} P_{V(\mathcal{T})}(\theta) = \sum_{D' \in \mathcal{K}, \varphi(D')=D} P_{\mathcal{K}}(D').$$

The next proposition is obvious.

PROPOSITION 16.5.6. *The function $P_{\mathcal{L}}$ is a discrete probability measure on \mathcal{L} .*

Define sets

$$\mathcal{L}_i = \varphi(\mathcal{K}_i) = \{\varphi(D(\theta)) \mid \theta \in V_{Q_i}\}$$

and functions $P_{\mathcal{L}_i} : \mathcal{L}_i \rightarrow \mathbb{R}$ such that for $D \in \mathcal{L}_i$,

$$P_{\mathcal{L}_i}(D) = \sum_{\theta \in V_{Q_i} \text{ and } \varphi(D(\theta))=D} P_{V(\mathcal{T})}(\theta).$$

PROPOSITION 16.5.7. *The function $P_{\mathcal{L}_i}$ is a discrete probability measure on \mathcal{L}_i .*

Notice that if the sets \mathcal{L}_i form a partition of \mathcal{L} then one can define an asymptotic density of diagrams from \mathcal{L} as follows. If $S \subseteq \mathcal{L}$ then

$$\rho(S) = \lim_{i \rightarrow \infty} P_{\mathcal{L}_i}(\mathcal{L}_i \cap S)$$

(when exists) is an asymptotic density of S in \mathcal{L} .

16.5.3. Iterative random generator RG_n . In this section we give an example of a complete sequence of termination conditions $\mathcal{Q} = \{Q_n\}_{n \in \mathbb{N}}$ and show that the corresponding probabilities $P_{\mathcal{K}_n}$ are related to a specific random diagram generator RG_n . We freely use notation from Section 16.5.2.

Let $Q_n : \Lambda \rightarrow \mathbb{N}$ ($n \in \mathbb{N}$) be a sequence of constant functions,

$$(38) \quad Q_n(\lambda) = n,$$

for every $\lambda \in \Lambda$. Clearly, Q_n is a random variable on Λ . So, one can view Q_n as a termination condition from Section 16.5.2. It follows that $\mathcal{Q} = \{Q_n\}_{n \in \mathbb{N}}$ is a complete sequence of termination conditions for Λ . Let $P_{\mathcal{K}_n}$ be the probability measure on V_{Q_n} from Section 16.5.2. One can describe the probability measure $P_{\mathcal{K}_n}$ in terms of the following random generator.

ALGORITHM 16.5.8 (Random Generator RG_n relative to the basic generator B).

INPUT: A presentation $\langle X; R \rangle$, a diagram $D_0 \in \mathcal{K}$, and $n \in \mathbb{N}$.

OUTPUT: A diagram from \mathcal{K} .

INITIALIZATION: Put $D_{m_0} = D_0$.

COMPUTATIONS:

- 1) Consequently compute $D_{m_{i+1}} = B(D_{m_i})$, for $i = 0, \dots, n - 1$.
- 2) Output D_{m_n} .

It is easy to see that the following assertion is true.

PROPOSITION 16.5.9. *Let $D \in \mathcal{K}$. Then*

$$P_{\mathcal{K}_n}(D) = \sum_{\theta \in V_{Q_n}, D(\theta)=D} P_{Q_n}(\theta)$$

is the probability of the event that D will be generated by RG_n .

REMARK 16.5.10. In a similar way one can construct a random generator to produce diagrams from \mathcal{L}_n . Indeed, apply RG_n to produce a diagram D from \mathcal{K} and take $\varphi(D)$. The corresponding probability to generate a diagram $\varphi(D)$ is equal to $P_{\mathcal{L}_n}(\varphi(D))$.

16.5.4. Diagram complexity and random generator RG_χ . In this section we define a notion of a size χ of a diagram and describe a random generator RG_χ which terminates when the diagram reaches a particular size.

For a diagram $D \in \mathcal{K}$ denote by $\chi_e(D)$ and $\chi_c(D)$, correspondingly, the number of *free edges* (i.e. edges which do not belong to the boundary of any cell in D) and the number of cells in D . We refer to the sum $\chi(D) = \chi_e(D) + \chi_c(D)$ as to the *size* of D .

Now, suppose that the basic extension B satisfies the following conditions for every $D \in \mathcal{K}$:

$$(39) \quad 0 \leq \chi(B(D)) - \chi(D) \leq 1;$$

$$(40) \quad \limsup_{i \rightarrow \infty} \chi(D(\lambda_i)) = \infty, \text{ for each } \lambda \in \Lambda;$$

$$(41) \quad \chi(D_0) = 0.$$

Under these assumptions on B we define the following random variables X_n and \hat{Q}_n . Recall that each $\theta \in V(\mathcal{T})$ is a path $e_1 \dots e_k$ in T_B with the origin at D_0 and the terminus at some diagram $D(\theta)$. Define a function $X_n : \Lambda \rightarrow \mathbb{N}$ by

$$X_n(\lambda) = \min\{i \in \mathbb{N} \mid \chi(D(\lambda_i)) = n\}$$

and put

$$(42) \quad \hat{Q}_n(\lambda) = \max\{i \in \mathbb{N} \mid \chi(D(\lambda_i)) = n\} = X_{n+1}(\lambda) - 1.$$

LEMMA 16.5.11. *Let B be a basic extension satisfying conditions (39), (40) and (41). Then X_n and \hat{Q}_n are random variables.*

Proof. It suffices to show that X_n is a random variable for every $n \in \mathbb{N}$, since $\hat{Q}_n = X_{n+1} - 1$. Fix arbitrary $n, j \in \mathbb{N}$. We claim that $X_n^{-1}(j)$ is a union of at most a countable number of cones.

Observe first that if $\lambda \in X_n^{-1}(j)$ then $Cone(\lambda_j) \subseteq X_n^{-1}(j)$. This implies that

$$X_n^{-1}(j) = \bigcup_{\lambda \in X_n^{-1}(j)} Cone(\lambda_j).$$

It is easy to see that the set above is either countable or finite union of cones and, hence, is measurable. ■

COROLLARY 16.5.12. *$\{\hat{Q}_i\}_{n \in \mathbb{N}}$ are termination conditions on Λ .*

Lemma 16.5.11 allows one to define probability spaces $(V_{\hat{Q}_i}, P_{\hat{Q}_i})$ (described in Section 16.5.2) for each $n \in \mathbb{N}$. We show in Proposition 16.5.14 below that the probability function $P_{\hat{Q}_i}$ can be described in terms of the following random generator.

ALGORITHM 16.5.13 (Random Generator RG_χ).

INPUT: A presentation $\langle X; R \rangle$, a diagram D_0 , and $n \in \mathbb{N}$.

OUTPUT: A diagram D such that $\chi(D) = n$.

INITIALIZATION: Put $D_{m_0} = D_0$ and $i = 1$.

COMPUTATIONS:

- 1) Construct $D_{m_i} = B(D_{m_{i-1}})$.
- 2) If $\chi(D_{m_i}) = n + 1$ then return $D_{m_{i-1}}$. Otherwise increment i and goto 1.

The next proposition is analogous to Proposition 16.5.9.

PROPOSITION 16.5.14. *Let \hat{Q}_n be defined as in (42) and $D \in \mathcal{K}$. Then*

$$P_{\mathcal{K}_n}(D) = \sum_{\theta \in V_{\hat{Q}_n}, D(\theta) = D} P_{\hat{Q}_n}(\theta)$$

is the probability of the event that D is generated by RG_χ .

REMARK 16.5.15. In a similar way one can construct a random generator to produce diagrams from \mathcal{L}_n . Indeed, apply RG_χ to produce a diagram D from \mathcal{K} and take $\varphi(D)$. The corresponding probability to generate a diagram $\varphi(D)$ is equal to $P_{\mathcal{L}_n}(\varphi(D))$.

Next, we define probability function $P_{V(\mathcal{T})}$ on $V(\mathcal{T})$ relative to the sequence of random variables $\{\hat{Q}_i\}_{i \in \mathbb{N}}$ (as in Section 16.5.2). To do this we need the following lemma.

LEMMA 16.5.16. *Assume that the basic extension B satisfies (39), (40), (41) and also satisfies an extra condition:*

$$(43) \quad \forall D_s \in \mathcal{K} \exists D_t = B(D_s) \text{ s.t. } \chi(D_t) - \chi(D_s) = 1.$$

Then $V(\mathcal{T}) = \bigcup_{i \in \mathbb{N}} V_{\hat{Q}_i}$ is a partition of $V(\mathcal{T})$.

Proof. By definition of \hat{Q}_i if $\theta \in V_{\hat{Q}_i}$ then $\chi(D(\theta)) = i$. Now, let $\theta \in V(\mathcal{T})$, $\chi(D(\theta)) = i$, and $D_s = D(\theta)$. By assumption of the lemma there exists $D_t = B(D_s)$ such that $\chi(D_t) - \chi(D_s) = 1$. Let (θ, θ') be the edge in \mathcal{T} , where $D(\theta') = D_t$. Then, by definition of \hat{Q}_i , $T_{\hat{Q}_i}(\text{Cone}(\theta')) = \{\theta\}$ and $\theta \in V_{\hat{Q}_i}$. Therefore $V_{\hat{Q}_i} = \{\theta \mid \chi(D(\theta)) = i\}$ and $V(\mathcal{T}) = \bigcup_{i \in \mathbb{N}} V_{\hat{Q}_i}$ is a partition of $V(\mathcal{T})$, as required. ■

COROLLARY 16.5.17. *$\{\hat{Q}_i\}_{n \in \mathbb{N}}$ is a complete system of termination conditions on Λ .*

COROLLARY 16.5.18. *Assume that the basic extension B satisfies all conditions (39)-(41) and (43). Then the following holds:*

- (1) *If $S \subseteq V(\mathcal{T})$ then*

$$\rho_\chi(S) = \lim_{i \rightarrow \infty} P_{\hat{Q}_i}(S \cap V_{\hat{Q}_i})$$

defines an asymptotic density of a set S in $V(\mathcal{T})$ with respect to $\{\hat{Q}_i\}$. We refer to ρ_χ as to the asymptotic density relative to the size of diagrams.

- (2) *If μ is a probability distribution on \mathbb{N} then one can define the discrete probability $P_{V(\mathcal{T})}$ on $V(\mathcal{T})$ as in (37).*
- (3) *$\mathcal{K} = \bigcup \mathcal{K}_i$ is a partition of \mathcal{K} and if $\mathcal{K}' \subseteq \mathcal{K}$ then*

$$\rho_\chi(\mathcal{K}') = \lim_{i \rightarrow \infty} P_{\mathcal{K}_i}(\mathcal{K}' \cap \mathcal{K}_i)$$

defines an asymptotic density of \mathcal{K}' in \mathcal{K} .

- (4) *Furthermore, if $\mathcal{L} = \mathcal{L}_i$ is a partition of \mathcal{L} (e.g. when φ preserves the size of diagrams) then*

$$\rho(\mathcal{L}') = \lim_{i \rightarrow \infty} P_{\mathcal{L}_i}(\mathcal{L}' \cap \mathcal{L}_i)$$

defines an asymptotic density of $\mathcal{L}' \subseteq \mathcal{L}$.

16.6. Basic extension algorithm B_S and relative probability measures

In this section we define a particular basic extension B_S , where S is a set of parameters. In the next section we use B_S to study asymptotic properties of diagrams.

16.6.1. Basic extension B_S . Let $\langle X; R \rangle$ be a finite presentation. Denote by $\mathcal{L} = \mathcal{L}(X, R)$ a set of representatives (up to isomorphism) of all diagrams D over $\langle X; R \rangle$. In this section we construct a particular basic extension B_S . Roughly speaking, B_S randomly adds cells and edges to the given diagram, and performs random foldings. The extension B_S depends on a set of parameters S (the probabilities with which it adds cells or edges to a diagram and makes foldings) which allow one to obtain random diagrams with different properties.

Let D be a diagram. Suppose a subset $M(D)$ of the set $V(D)$ of vertices of D is chosen. The vertices from $M(D)$ are called “marked vertices” (worked out vertices). Suppose also that a subset $A(D) \subseteq \partial D \setminus M(D)$ of non-marked vertices from D is chosen such that $|A(D)| \leq 1$. We refer to vertices from $A(D)$ as to “active vertices” (vertices in the working). The triple $(D, M(D), A(D))$ is called an *extended diagram*. Morphisms of extended diagrams are morphisms of diagrams that preserve the marked and active vertices. Let $\mathcal{K} = \mathcal{K}(X, R)$ be the set of all extended diagrams from \mathcal{L} .

Let $S = (s_1, s_2, s_3, s_4)$ be a sequence of reals such that $s_i \in [0, 1]$ and $s_1 + s_2 + s_3 = 1$. The following procedure provides the basic extension B_S .

ALGORITHM 16.6.1 (Basic extension B_S).

INPUT: Let D be an extended van Kampen diagram over $\langle X; R \rangle$ such that either $A(D) \neq \emptyset$ or $\partial D - M(D) \neq \emptyset$.

OUTPUT: Diagram $B_S(D) = D^*$.

- 1) If $|A(D)| = 1$ then take the only vertex $v \in A(D)$. If $|A(D)| = 0$ then choose randomly and uniformly an unmarked vertex $v \in \partial D - M(D)$ and put $A(D) = \{v\}$.
- 2) If v is not the last unmarked vertex in ∂D then with probability s_1 do a), with probability s_2 do b), and with probability $s_3 = 1 - s_1 - s_2$ do c) below:
 - a) Take randomly and uniformly a relator $r \in R$. Make a face N with the boundary label r at some vertex $u \in \partial N$. Attach N to v by identifying v with u . Go to 5).
 - b) Generate randomly and uniformly a letter $y \in X^{\pm 1}$. Make a free edge $e = (u_1, u_2)$ with the label y . Attach e to v by identifying v with u_1 . Go to 5).
 - c) Do not attach anything to v and go to 4).
- 3) If v is the last unmarked vertex in ∂D and $s_1 + s_2 \neq 0$ then with probability $\frac{s_1}{s_1 + s_2}$ do a) below, otherwise do b):
 - a) Take randomly and uniformly a relator $r \in R$. Make a face N with the boundary label r at some vertex $u \in \partial N$. Attach N to v by identifying v with u . Go to 5).
 - b) Generate randomly and uniformly a letter $y \in X^{\pm 1}$. Make a free edge $e = (u_1, u_2)$ with the label y . Attach e to v by identifying v with u_1 . Go to 5).
- 4) Let $(e_1, h_1), \dots, (e_k, h_k)$ be all pairs of edges incident to v and such that for each i the following conditions hold:
 - the path $e_i h_i$ belongs to the boundary of the diagram (with respect to a fixed orientation);
 - all endpoints of e_i and h_i are unmarked;
 - e_i and h_i^{-1} have the same labels (potential fold);
 - edges e_i and h_i are not free.
 Then:
 - a) For each $i = 1, \dots, k$ with a fixed probability s_4 fold e_i and h_i^{-1} .
 - b) Mark v , and add it to $M(D)$.
 - c) Remove v from $A(D)$. Go to 5).
- 5) Denote the resulting diagram by $B_S(D)$. Output $B_S(D)$.

Below we list some properties of B_S . Recall that a vertex v is a *cut vertex* of a map \mathcal{M} if there exist two vertices $v_1, v_2 \in \mathcal{M}$ such that any path connecting v_1 and v_2 goes through v .

LEMMA 16.6.2. (Properties of B_S) Let $D^* = B_S(D)$. Then:

- 1) If a vertex v in D is marked then $N_D(v) = N_{D^*}(v)$.
- 2) If every vertex $v \in D - \partial D$ is marked then every vertex $v \in D^* - \partial D^*$ is marked.
- 3) If every cut vertex in D is either marked or active then every cut vertex in D^* is either marked or active.
- 4) Given a diagram D there are only finitely many possible outcomes for D^* .
- 5) If $\partial D - M(D) \neq \emptyset$ then $\partial D^* - M(D^*) \neq \emptyset$.

Proof. Follows from the description of B_S . ■

Let D be a diagram. The following notation,

$$D^* = B_S^{(n)}(D),$$

means that D^* is a result of n applications of B_S to the diagram D .

REMARK 16.6.3. Let $D^* = B_S^{(n)}(D_0)$, where D_0 is a diagram which consists of one vertex. Then:

- 1) If $s_2 = 1$ then D^* is a tree.
- 2) If $s_1 = 1$ then D^* is a “tree of cells”, i.e., a diagram without free edges and such that the dual graph is a tree.
- 3) If $s_1 + s_2 = 1$ then D^* is a “tree of cells and free edges”.

Let \mathcal{W}_S be the random walk corresponding to the random generator RG_S . In the following lemma we collect some basic properties of \mathcal{W}_S . By D_∞ we denote a path (perhaps, infinite) in the graph T :

$$D_1 \rightarrow D_2 \rightarrow \dots \rightarrow D_k \rightarrow \dots$$

LEMMA 16.6.4 (Properties of T). If D_∞ is a path in T and $D_i \in D_\infty$ then the following hold:

- 1) every vertex $v \in D_i - \partial D_i$ is marked;
- 2) for every marked vertex $v \in D_i$ the neighborhood of v does not change in D_j for $j \geq i$, i.e., $N_{D_j}(v) = N_{D_i}(v)$;
- 3) every unmarked vertex $v \in D_i$ either stays unmarked in all D_j for $j \geq i$ or eventually it becomes active;
- 4) every active vertex $v \in D_i$ either stays active in all D_j for $j \geq i$ (and in this event the case 4) in the description of B_S does not occur) or eventually it becomes marked.

COROLLARY 16.6.5. The random basic extension B_S is locally stable at every marked vertex v .

16.6.2. Completeness of the basic extension B_S . In this section we show that B_S is \mathcal{L} -complete provided none of the probabilities in S are zero.

Below we use notation from the previous sections. Recall that $\mathcal{L} = \mathcal{L}(X, R)$ is a set of representatives of all van Kampen diagrams over $\langle X; R \rangle$ up to isomorphisms and $\Phi = \Phi_{B_S}(D_0)$ is the set of all extended diagrams over $\langle X; R \rangle$ that can be produced by a sequence of applications of B_S starting from D_0 .

For an extended diagram D denote by \overline{D} the ordinary diagram that results from D by erasing the sets $M(D)$ and $A(D)$. Slightly abusing notation we will identify vertices, edges, and cells in D and \overline{D} . This implies, in particular, that for a vertex $v \in D$ one has $N_D(v) = N_{\overline{D}}(v)$. In the situations when D is an extended diagram and $\varphi : \overline{D} \rightarrow C$ is a morphism of ordinary diagrams the agreement above will allow us to consider unambiguously the image $\varphi(v)$ for a vertex $v \in D$. Put

$$\overline{\Phi}_{B_S} = \{\overline{D} \mid D \in \Phi\}.$$

According to the definition of completeness from Section 16.5.1 the basic extension B_S is \mathcal{L} -complete relative to the mapping $\overline{-} : \Phi_{B_S} \rightarrow \mathcal{L}$ if $\overline{\Phi}_{B_S} = \mathcal{L}$. For notational convenience, further in this section we omit the index B_S in $\overline{\Phi}_{B_S}$ and denote $\overline{\Phi}_{B_S}$ simply by $\overline{\Phi}$.

In the proof of completeness we use two auxiliary transformations of maps termed *edge cuts* and *vertex cuts*. First, we define the edge cut transformation. Let \mathcal{M} be an arbitrary map over $\langle X; R \rangle$ and $e = (v, u)$ be an edge from $\mathcal{M} - \partial\mathcal{M}$ with $v \in \partial\mathcal{M}$. Let f_1 and f_2 be two edges in $\partial\mathcal{M}$ incident to v such that there are no cells in $N(v)$ and no edges in $\text{Star}(v)$ between f_1 and f_2 according to the fixed orientation (see Section 16.3.2). Then the cut of the edge e between f_1 and f_2 is the following sequence of transformations:

- replace (“cut”) the vertex v and the edge e with their two copies v_1, v_2 and $e_1 = (v_1, u), e_2 = (v_2, u)$;
- replace the vertex v in all edges and cells in \mathcal{M} between e and f_1 , including f_1 , with v_1 ;
- replace the vertex v in all edges and cells in \mathcal{M} between f_2 and e , including f_2 , with v_2 .

Figure 16.4 illustrates an edge cut.

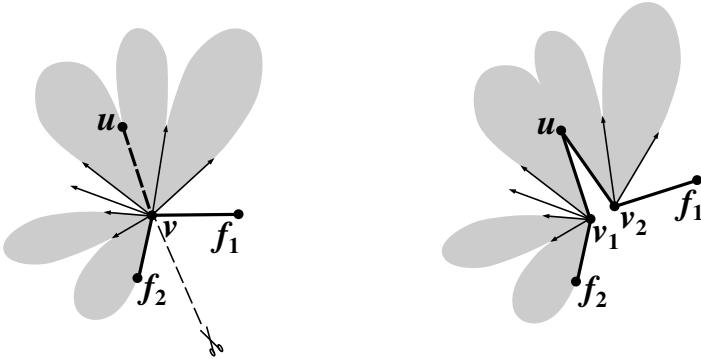


FIGURE 16.4. Edge cut.

Now we define the vertex cut transformation. Let v be a vertex on $\partial\mathcal{M}$ and (f_1, f_2) and (g_1, g_2) be pairs of distinct edges from $\partial\mathcal{M}$ incident to v such that there are no cells from $N(v)$ and no edges in $\text{Star}(v)$ between f_1 and f_2 and, also, between g_1 and g_2 (according to the fixed orientation). Then the cut of v between (f_1, f_2) and (g_1, g_2) is the following sequence of transformations:

- replace (“cut”) the vertex v with two copies v_1, v_2 ;

- replace the vertex v in all edges and cells in \mathcal{M} between f_2 and g_1 , including f_2 and g_1 , with v_1 ;
- replace the vertex v in all edges and cells in \mathcal{M} between g_2 and f_1 , including g_2 and f_1 , with v_2 .

Figure 16.5 illustrates a vertex cut. We allow only those vertex cuts that result in a connected map.

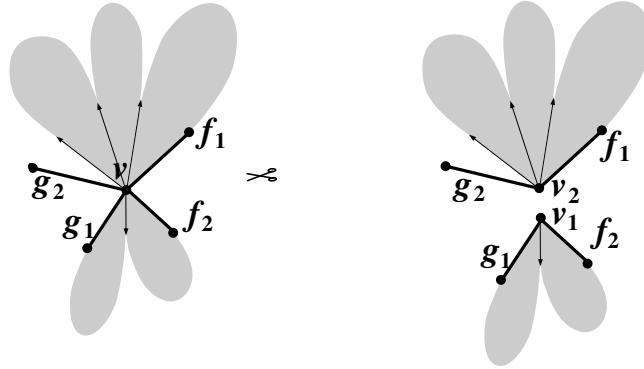


FIGURE 16.5. Vertex cut.

We refer to these vertex and edge cuts as *simple cuts*. If σ is a simple cut of \mathcal{M} which results in a map \mathcal{M}' then we write $\mathcal{M} \xrightarrow{\sigma} \mathcal{M}'$. There exists a natural *sewing* morphism $\varphi : \mathcal{M}' \rightarrow \mathcal{M}$ which sews \mathcal{M}' back into \mathcal{M} (φ identifies v_1 with v_2 and e_1 with e_2 from the definitions above). We say that a map \mathcal{M}' is a *cut* of \mathcal{M} if \mathcal{M}' can be obtained from \mathcal{M} by a sequence of simple cuts $\sigma = (\sigma_1, \dots, \sigma_k)$. We allow here the empty sequences too (i.e., \mathcal{M} is a cut of itself). If \mathcal{M}' is a cut of \mathcal{M} then there exists a natural *sewing* morphism $\mathcal{M}' \rightarrow \mathcal{M}$ which is a composition of the sewing morphisms corresponding to the sequence of simple cuts from \mathcal{M}' to \mathcal{M} .

THEOREM 16.6.6. *Let $\langle X; R \rangle$ be a symmetrized finite presentation and D be a van Kampen diagram over $\langle X; R \rangle$ which*

- 1) *does not contain loops of length 1 and*
- 2) *for each cell c the boundary ∂c is vertex-simple (does not touch itself).*

If none of the probabilities in S is zero then $D = \overline{B_S^n(D_0)}$ for some $n \in \mathbb{N}$, i.e., the diagram D can be generated by RG with non-trivial probability in n iterations.

Proof. Let $D \in \mathcal{L}$ be a van Kampen diagram. We are going to construct by induction a sequence of extended diagrams

$$D_0, \dots, D_n$$

and a sequence of morphisms

$$\varphi_0 : \overline{D_0} \rightarrow D, \dots, \varphi_n : \overline{D_n} \rightarrow D$$

such that $\varphi_n : \overline{D_n} \rightarrow D$ is an isomorphism and such that the following conditions hold:

- P1) D_i can be obtained from D_{i-1} by a sequence of basic extensions of the following type:

- a) choose some unmarked vertex v in D_{i-1} and make it active;
 - b) add finitely many (perhaps zero) cells and free edges to D_{i-1} at v ;
 - c) fold some edges incident to v ;
 - d) make v marked and non-active.
- P2) \overline{D}_i is a cut of the subcomplex $\varphi_i(\overline{D}_i)$ of D and $\varphi_i : \overline{D}_i \rightarrow \varphi_i(\overline{D}_i)$ is the corresponding sewing morphism.
- P3) For every marked vertex $v \in D_i$, φ_i maps the neighborhood $N_{D_i}(v)$ of v bijectively on the neighborhood $N_D(\varphi_i(v))$.

Base of induction $i = 0$. Recall that D_0 is an extended diagram consisting of a single unmarked vertex v and such that $M(D_0) = \emptyset$, $A(D_0) = \emptyset$. Take any vertex v_0 in D and define $\varphi_0(v) = v_0$. All properties P1-P3 clearly hold for D_0 .

Induction step. Let D_i and φ_i satisfying properties P1-P3 have been constructed.

In the following claims we study properties of \overline{D}_i , $\varphi_i(\overline{D}_i)$, and φ_i . Recall that by $FE(D)$ and $C(D)$ we denote sets of free edges and cells of D .

Claim 1 (Properties of D_i). *The following hold.*

- 1.1) Every two elements of $FE(D_i) \cup C(D_i)$ are connected by a chain (see Section 16.3.3 for definitions) of marked vertices (marked chain).
- 1.2) Cut vertices of D_i are marked.
- 1.3) Every element of $FE(D_i) \cup C(D_i)$ has a marked vertex.

Proof of Claim 1. Obviously, 1.2) and 1.3) are corollaries of 1.1).

We prove 1.1) by induction on i . For $i = 0$ there is nothing to prove. Assume now that 1.1) holds for $i = l$. By the property P1, D_{l+1} can be obtained from D_l first by choosing an unmarked vertex v , adding free edges and cells to v , folding some edges incident to v , and making v marked and non-active. Clearly, every new cell or free edge in $D_{l+1} - D_l$ contains the marked vertex v which connects them to the rest of the diagram. \square

$\varphi_i(\overline{D}_i)$ is a map on a plane \mathbb{R}^2 . Hence $\mathbb{R}^2 - \varphi_i(\overline{D}_i)$ is a disjoint union of open, connected, simply connected components C_0, \dots, C_m , where C_0 is the unbounded component, and all other components are bounded. By ∂C_s we denote the boundary of C_s which is a connected component of $\partial \varphi_i(\overline{D}_i)$.

Claim 2 (Properties of $\varphi_i(D_i)$). *The following hold.*

- 2.1) If u is a vertex in D_i and $\varphi_i(u) \in \partial C_s$ for some finite component C_s ($s > 0$) then u is unmarked in D_i .
- 2.2) If $e = (u, v)$ is an edge in ∂C_s ($s > 0$) then there exists a cell $f \in \varphi_i(D_i)$ such that $e \in \partial f$.

Proof of Claim 2.

2.1) Let $u \in D_i$ and $\varphi_i(u) \in \partial C_s$. Since $\varphi_i(u) \in \partial C_s$ the morphism φ_i is not bijective on $N_{D_i}(u)$. Hence u is unmarked by the P3.

2.2) Assume that $e = (v, u) \in \partial C_s$ ($s > 0$). Assume e does not belong to the boundary of any cell of $\varphi_i(D_i)$. Let $e' = (v', u') \in D_i$ be such that $\varphi_i(e') = e$. The vertices v' and u' are unmarked in D_i by 2.1). Clearly, the edge e' does not belong to the boundary of any cell of D_i , because D_i is a cut of $\varphi_i(D_i)$. Hence e' is a free edge in D_i . By the property 1.3 at least one of its endpoints is marked. Contradiction. \square

By the property P2 \overline{D}_i is obtained from the subcomplex $\varphi_i(\overline{D}_i)$ of D by a finite sequence of simple cuts $\sigma = (\sigma_1, \dots, \sigma_k)$:

$$(44) \quad \varphi_i(\overline{D}_i) = B_1 \xrightarrow{\sigma_1} B_2 \xrightarrow{\sigma_2} \dots \xrightarrow{\sigma_k} B_{k+1} = \overline{D}_i.$$

Notice that $\varphi_i(\partial\overline{D}_i) = \partial\varphi_i(\overline{D}_i) \cup \Gamma$, where Γ is the set of edges of $\varphi_i(D_i)$ that were cut in (44). Edges from Γ have exactly two preimages in ∂D_i and edges from $\partial\varphi_i(\overline{D}_i)$ have unique preimages. Since D_i is a diagram it is connected, hence ∂D_i is a closed path, as well as $\varphi_i(\partial\overline{D}_i)$. Let $e_0 = (v_0, v') \in \partial D_i$ be an edge such that $\varphi_i(e_0) \in \partial C_0$. Denote by P the closed path $\varphi_i(\partial\overline{D}_i)$ starting with the edge $\varphi_i(e_0)$.

Claim 3 (Properties of P). *Let C_s and C_t be two distinct components. Let e_1, \dots, e_p be edges of ∂C_s , and d_1, \dots, d_r be edges of ∂C_t both given in the order they appear in P . Then the following holds:*

- 1) *The boundary ∂C_s , as a path, is a cyclic permutation of e_1, \dots, e_p .*
- 2) *The boundary paths ∂C_s and ∂C_t appear in P in one of the following orders:*

$$(45) \quad e_1, \dots, e_{p'}, d_1, \dots, d_r, e_{p'+1}, \dots, e_p$$

or

$$(46) \quad d_1, \dots, d_{r'}, e_1, \dots, e_p, d_{r'+1}, \dots, d_r,$$

where $0 \leq p' \leq p$ and $0 \leq r' \leq r$.

Proof of Claim 3. Induction on the number of simple cuts in (44). Notice that path P might be not simple, it can be a union of disjoint simple loops. If the number of cuts is zero then φ_i is an isomorphism, so there is only one component, the infinite component C_0 , and the claim is obvious.

Assume the claim is true for $k - 1$ simple cuts $\sigma_1, \dots, \sigma_{k-1}$. Suppose now that σ_k is a cut along some edge $f = (v, u)$. Then $v \in C_j$ for some $0 \leq j \leq q$ and $f \notin \partial C_l$ for any component C_l . Recall that a cut along f replaces the edge e by two new edges f_1 and f_2 incident to u in such a way that the boundary ∂B_{k+1} is obtained from ∂B_k by inserting either a path $f_1^{-1}f_2$, or a path $f_2^{-1}f_1$ (depending on the orientation). Assume that the path $f_1^{-1}f_2$ was inserted. If $j \neq s$ and $j \neq t$ then the claim follows by induction. We may assume now that $j = s$. The image of $f_1^{-1}f_2$ in B_k under the sewing morphism $\theta : B_{k+1} \rightarrow B_k$ is the path $f^{-1}f$. Observe that f does not belong to components of B_k . Now the claim follows by induction since $\varphi_i(D_i)$ is obtained from B_k by a sequence of a sewing morphisms. This case is illustrated on Figure 16.6.

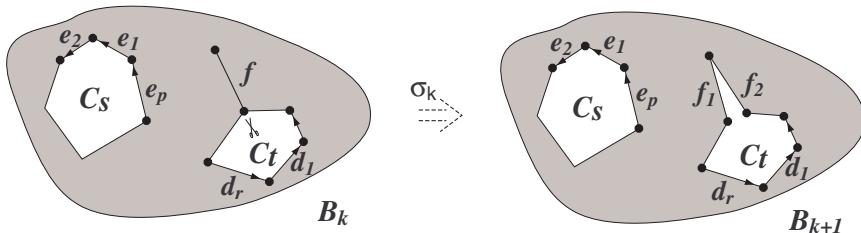


FIGURE 16.6. Edge cut starting from the finite component C_t .

Assume now that σ_k is a vertex cut. This case is depicted on Figure 16.7. Here two components of B_k are merged into one component in B_{k+1} . The proof is obvious from the picture and we omit it. \square

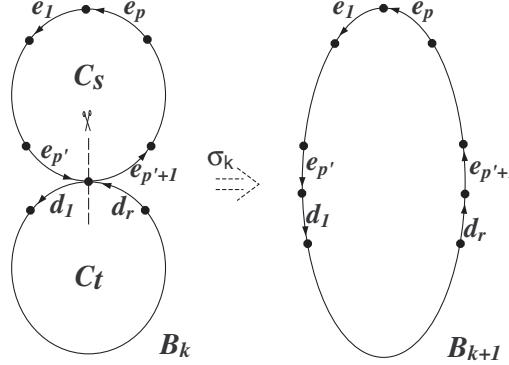


FIGURE 16.7. Vertex cut starting from the finite component C_t .

It follows from Claim 3 that if P contains a subpath $e_1P_1e_2$ such that $e_1, e_2 \in \partial C_s$ for some component C_s and a subpath P_1 does not contain edges from C_s then:

- The terminus of e_1 is the origin of e_2 .
- If P_1 contains an edge $e_3 \in \partial C_t$ for another component C_t then P_1 contains all edges from ∂C_t .

A path Q in $\varphi_i(D_i)$ is called *component-complete* provided, if Q contains an edge from ∂C_t , then Q contains all edges from ∂C_t .

Now we define a directed graph (actually, a forest) $T = T(D_i, \varphi_i)$ related to the cut (44). The set of vertices of T is the set of components C_0, \dots, C_q . Two components C_s and C_t are connected by an edge from C_s to C_t if and only if $C_s \neq C_t$ and P contains a subpath $e_1P_1fP_2e_2$ such that

- a) $e_1, e_2 \in \partial C_s$;
- b) P_1 is component-complete;
- c) $f \in \partial C_t$.

It follows from Claim 3 that T does not contain cycles, hence it is a forest. See Figure 16.8 for an example of a graph T .

Claim 4 (Existence of an unmarked vertex with a unique image). *If $\varphi_i : \overline{D}_i \rightarrow D$ is not an isomorphism then there exists an unmarked vertex $v \in \overline{D}_i$ such that $\varphi_i(v) \neq \varphi_i(v')$ for every $v' \in \overline{D}_i$ with $v \neq v'$.*

Proof of Claim 4. Recall that k is the number of simple cuts in (44). We consider two cases: $k = 0$ and $k > 0$.

CASE 1. Let $k = 0$. Then the sewing morphism $\varphi_i : \overline{D}_i \rightarrow \varphi_i(\overline{D}_i)$ is an isomorphism. If $\varphi_i(\overline{D}_i) = D$ then we have nothing to prove. If $\varphi_i(\overline{D}_i) \neq D$ then there exists a vertex $u \in \varphi_i(\overline{D}_i) \subset D$ such that $N_D(u) \neq N_{\varphi_i(\overline{D}_i)}(u)$. Therefore, if $v = \varphi_i^{-1}(u) \in D_i$ then $N_D(\varphi_i(v)) \not\simeq N_{\varphi_i(\overline{D}_i)}(v)$ (since the latter one is isomorphic to $N_{\varphi_i(\overline{D}_i)}(\varphi_i(v))$). Now v is unmarked by the property P3. Clearly, $\varphi_i(v') \neq \varphi_i(v)$ for any $v' \neq v$, since φ_i is an isomorphism.

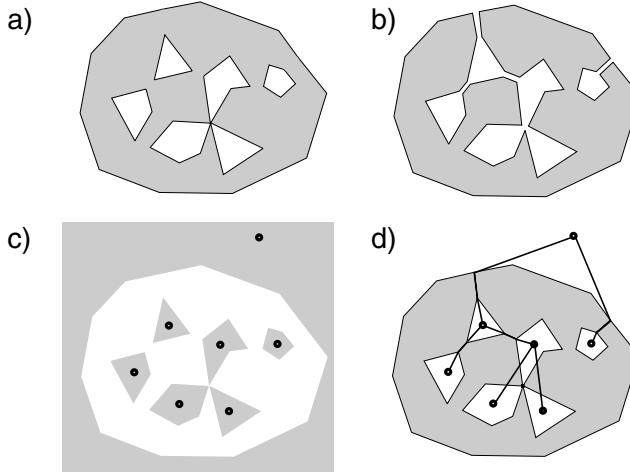


FIGURE 16.8. Example to Case 2.2: a) $\varphi_i(\overline{D}_i)$, b) \overline{D}_i , c) complement set $\mathbb{R}^2 - \varphi_i(\overline{D}_i)$ with components marked by points, d) the graph $T(D_i, \varphi_i)$ shown on $\varphi_i(\overline{D}_i)$.

CASE 2. Let $k \geq 1$. We say that an edge $e \in \varphi_i(\overline{D}_i)$ is cut by a *terminal edge cut* if there exist two edges $e_1 = (u_1, v)$ and $e_2 = (u_2, v)$ in \overline{D}_i mapped to e by φ_i . In this event the vertex v is not cut by a simple cut between edges e_1 and e_2 . Observe that if the sequence of cuts (44) does not contain a vertex cut then there exists a terminal edge cut.

CASE 2.1. Let $e = (\bar{u}, \bar{v}) \in \varphi_i(\overline{D}_i)$ be an edge split by a terminal edge cut to edges $e_1 = (u_1, v)$ and $e_2 = (u_2, v)$ in \overline{D}_i . We claim that v is a required vertex. Observe first that the vertex v is unmarked. Indeed, e was cut by a simple cut σ_s hence the sewing morphism φ_i is not bijective on $N_{\overline{D}_i}(v)$. Therefore v is unmarked by the property P3.

Assume now that there exists a vertex $v' \in \overline{D}_i$ such that $v' \neq v$ and $\varphi_i(v') = \varphi_i(v)$. Then v is a cut vertex in D_i . Indeed, in this event v is cut by a simple cut from (44). As we have mentioned above this simple cut is not a cut between edges e_1 and e_2 . It follows that there are two different pairs of edges in $Star_{D_i}(v)$ which do not have any cells or free edges in between (see Figure 16.9). Since D_i is simply connected v is a cut vertex. Then v is marked by 1.2 of Claim 1. Contradiction.

CASE 2.2. Assume that there is no edge in $\varphi_i(\overline{D}_i)$ cut by a terminal edge cut. Let $T = T(\overline{D}_i, \varphi_i)$ be the graph related to the cut (44). We claim that there exists at least one finite component in $\mathbb{R}^2 - \varphi_i(D_i)$. Indeed, otherwise there is no a vertex cut in (44) (vertex cuts require at least one finite component, since D_i is connected). Hence, as was mentioned above, there is a terminal edge cut in (44) which is not the case.

Let C_s be a leaf of T different from C_0 (there are at least two leafs in T , hence it exists). The map $\varphi_i(D_i)$ is a submap of the van Kampen diagram D . Hence ∂C_s is a loop in D . Therefore, by assumption of the theorem on D , has length at least 2. So there are at least two vertices on ∂C_s . We claim that there is a vertex on ∂C_s that was not cut by (44). To show this consider the path P . Let e_1, \dots, e_l be edges of ∂C_s in the order they appear in P . By Claim 3.1 e_1 and e_2 are consecutive

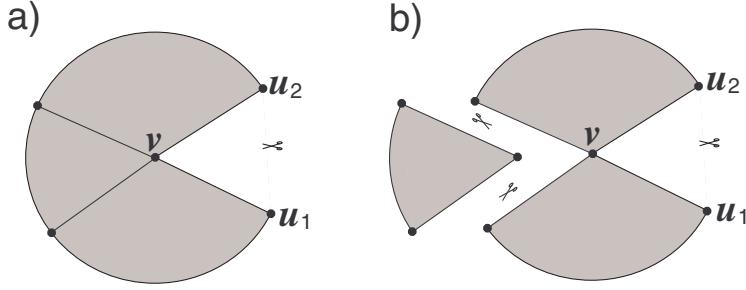


FIGURE 16.9. Example to Case 2.1: a) neighborhood of a rigid vertex of a split edge, b) neighborhood of a rigid vertex of a split edge with one piece cut off.

edges of ∂C_s . Denote by u the terminus of e_1 . If u was not cut then this is a vertex we are looking for. Assume now that u was cut. There are two cases here: either e_1 and e_2 are consecutive edges of P or they are not.

In the former case u was not cut in between e_1 and e_2 , so it was cut in between two other edges (see Figure 16.10). In this event there exists a preimage u' of u which is a cut vertex in D_i . By Claim 1.2) u' is marked, but it cannot be marked as a preimage of a vertex from ∂C_s . This contradiction shows that u is cut in between e_1 and e_2 .

Therefore, P contains a subpath $e_1 P_1 e_2$, where $P_1 = f_1 \dots f_l$ is a non-empty loop. Since C_s is a leave P_1 does not contain edges from boundaries ∂C_t . Hence P_1 consists only of edges from Γ . Observe that if P_1 does not contain a backtrack then it bounds a cell inside. Indeed, by definition for each edge e from Γ there are two different cells whose boundaries contain e . If $f_1 \neq f_1^{-1}$ then one of the cells which contain f_1 on their boundaries is bounded by P_1 . If $f_l = f_1^{-1}$ then the subpath f_2, \dots, f_{l-1} is a loop in Γ and induction finishes the proof. Now if P_1 bounds a cell inside then D_i is not connected. This contradiction shows that there is a backtrack in P_1 . Hence there is a terminal edge cut in (44), which is impossible. This shows that u was not cut and hence it has a unique preimage v with respect to φ_i . v is unmarked by Claim 2.1). This proves Claim 4. \square

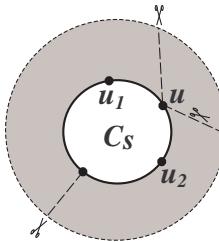


FIGURE 16.10. Non-cut vertex u on ∂C_s .

At this point we have (D_i, φ_i) satisfying properties P1-P3 and an unmarked vertex $v \in \overline{D}_i$ with the unique image in D . We are going to construct a new diagram

D_{i+1} and a morphism $\varphi_{i+1} : \overline{D}_{i+1} \rightarrow D$ satisfying properties P1-P3. Consider a subcomplex E_{i+1} of D generated by $\varphi_i(\overline{D}_i)$ and $N_D(\varphi_i(v))$ (it is connected, but might be not simply connected, so not a diagram). Let K_{i+1} be the subcomplex generated by elements from $N_D(\varphi_i(v))$ that do not belong to $\varphi_i(\overline{D}_i)$ (i.e. $K_{i+1} = N_D(\varphi_i(v)) - \varphi_i(\overline{D}_i)$).

Claim 5. *There exists an extended diagram D_{i+1} and a sewing morphism $\varphi_{i+1} : D_{i+1} \rightarrow E_{i+1}$ such that (D_{i+1}, φ_{i+1}) satisfies properties P1-P3.*

Proof of Claim 5. Choose v to be an active vertex and assign $A(D_i) = \{v\}$. Consider cases from the proof of Claim 4.

CASE 1. Let $k = 0$. Then the sewing morphism $\varphi_i : \overline{D}_i \rightarrow \varphi_i(\overline{D}_i)$ is an isomorphism. Since \overline{D}_i is a diagram, it is simply connected, so is $\varphi_i(\overline{D}_i)$. This implies that there are no finite components in $\mathbb{R}^2 - \varphi_i(\overline{D}_i)$, only the infinite one C_0 . The submap K_{i+1} is finite, so it cannot fill in the whole region C_0 . Hence there exists an edge $e \in \partial E_{i+1} \cap \partial K_{i+1}$ (see Figure 16.11).

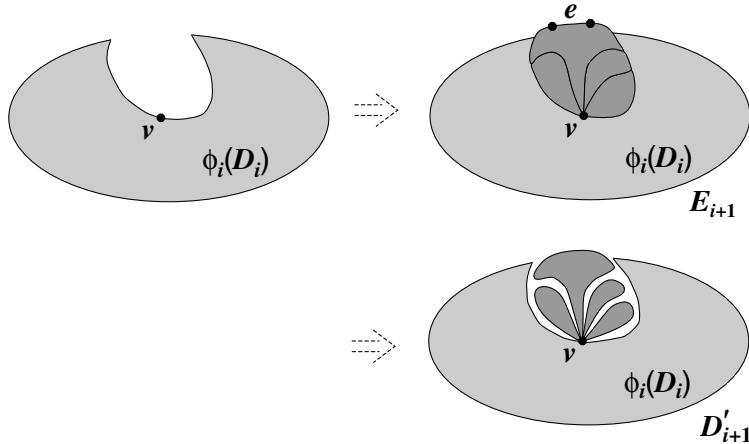


FIGURE 16.11. Diagrams $\varphi_i(D_i)$, E_{i+1} , and D'_{i+1} .

Now we make a sequence of edge cuts to cut E_{i+1} as follows. We cut all necessary edges in K_{i+1} (not cutting the vertex v) into a bouquet B of free edges and cells attached to the rest of the diagram at the vertex v . Since there exists at least one edge in K_{i+1} on the boundary ∂E_{i+1} we can start the cutting process. Since D does not contain loops of length 1 we can make it into a bouquet without cutting the vertex v . We denote by D'_{i+1} the complex resulted from the cutting of E_{i+1} . From the construction D'_{i+1} is union of \overline{D}_i and the bouquet B , hence it is a diagram.

Now, we fold edges in D'_{i+1} adjacent to v , if they are folded in E_{i+1} , and denote the result by D_{i+1} . Observe that the diagram D_{i+1} can be obtained from D_i as follows:

- choose the vertex v (which is unmarked and non-active) and make it active;
- attach finitely many cells and free edges to D_i at v ;

- c) fold some edges incident to v ;
- d) make v unmarked and non-active.

We claim that all these steps can be performed by the random extension B_S . The existence of the vertex v comes from Claim 4. Notice that in this case the vertex v was chosen in the proof of Claim 4 in such a way that $N_{\varphi_i(\overline{D}_i)}(\varphi_i(v))$ is a proper subcomplex of $N_D(\varphi_i(v))$. Addition of new free edges and cells is allowed in B_S since all the probabilities in S are non-zero. Therefore, even if v is the last unmarked vertex in D_i the extension B_S is allowed to add all required cells and free edges to D_i at v . Notice that any vertex $\partial K_{i+1} \cap \partial \overline{D}_i$ is unmarked. Hence, folds of edges between cells in K_{i+1} and D_i are allowed by B_S too. Therefore, D_{i+1} can be obtained from D_i by a finite number of application of B_S and the property P1 is satisfied.

D'_{i+1} , as well as D_{i+1} , is a cut of E_{i+1} which is a subcomplex of D . Define $\varphi_{i+1} : \overline{D}_{i+1} \rightarrow D$ to be the corresponding sewing morphism. Hence we have the property P2. Again it is clearly true (from the way we constructed D_{i+1}) that φ_i is an isomorphism of $N_{D_{i+1}}(v)$ onto $N_D(\varphi_{i+1}(v))$ which proves P3. This finishes the proof in Case 1.

CASE 2.1. Let $k > 0$ and the sequence (44) has a terminal edge cut, say σ_l . In this case there exists an edge $e = (\bar{u}, \bar{v}) \in \varphi_i(D_i)$ which is cut by σ_l into two edges $e_1 = (u_1, v)$ and $e_2 = (u_2, v)$ in \overline{D}_i . By Claim 4 (see Case 2.1) the vertex v is unmarked and $\varphi_i(v) \neq \varphi_i(v')$ for any vertex $v' \in \overline{D}_i$, so it can be chosen as the active vertex. It follows (see the proof of Claim 4, Case 2.1) that $K_{i+1} = \emptyset$ and the equality $\varphi_i(\overline{D}_i) = E_{i+1}$ holds. In this case we construct the diagram D_{i+1} from D_i as follows:

- a) choose the vertex v and make it active;
- b) fold edges e_1 and e_2 in D_i and make the vertex v marked and non-active.

Notice that the vertices u_1 and u_2 are unmarked in D_i . Since the probabilities from S are non-zero and there are unmarked vertices in D_i , besides v , it follows that B_S can add no cells and no free edges at $v \in D_i$. In this case B_S can fold edges e_1 and e_2 in D_i since endpoints of e_1 and e_2 are unmarked. So D_{i+1} satisfies the property P1. It follows from the construction that D_{i+1} is a cut of $E_{i+1} \subset D$. Let φ_{i+1} be the corresponding sewing morphism. Clearly the properties P2 and P3 holds.

CASE 2.2. Let $k > 0$ and there is no terminal edge cut in (44). Let C_s be a leave in a forest $T(D_i, \varphi_i)$ different from C_0 . By Claim 4 there is an unmarked vertex $v \in \partial C_s$ that can be chosen as the active vertex. In this case K_{i+1} is not trivial (since C_s is a finite component). The vertex v is not the only unmarked vertex in D_i since all vertices on ∂C_s must be unmarked and the number of vertices on ∂C_s is not less than 2.

In this case E_{i+1} is obtained from $\varphi_i(D_i)$ by adding some number of cells and free edges to the vertex v inside the component C_s . The sequence of cuts (44) cuts $\varphi_i(D_i)$ into \overline{D}_i starting (by definition) at some vertex from $\partial \varphi_i(D_i)$. The path P from Claim 3 starts at some edge $e_0 \in \partial C_0$. The path P naturally defines the sequence of cuts of $\varphi_i(D_i)$ which results in D_i . Since P starts on ∂C_0 there exists a sequence σ' of simple cuts starting at the boundary of C_0 and leading to the component C_s and stopping at a vertex w on ∂C_s . Moreover, we may assume that there are no simple cuts in σ' starting at vertices of ∂C_s (σ' corresponds to the initial path of P which reaches ∂C_s the first time). Then being on ∂C_s we can cut

K_{i+1} into a bouquet of cells and free edges at the vertex v . After that we can make all cuts required to produce \overline{D}_i from $\varphi_i(D_i)$ (it is possible because all the needed boundary vertices of $\varphi_i(D_i)$ are available to us now).

Denote the resulting diagram by D'_{i+1} . The diagram D'_{i+1} consists of \overline{D}_i with the bouquet of free edges and cells attached at v . Finally, we fold the edges incident to v that are folded in E_{i+1} , then make v marked and non-active. Denote the resulting diagram by D_{i+1} . The argument similar to the one in Case 2.1 shows that D_{i+1} satisfies all of the properties P1-P3. \square

By Claims 4 and 5 if $\varphi_i : \overline{D}_i \rightarrow D$ is not an isomorphism one can construct $\varphi_{i+1} : \overline{D}_{i+1} \rightarrow D$ such that D_{i+1} has strictly more marked vertices than D_i (we do not fold marked vertices from D_i when constructing D_{i+1}). By P2, D_i is a cut of $\varphi_i(D_i)$ which is a submap of D . Since every edge can be cut only once we get the bound

$$|E(D_i)| \leq 2|E(D)|$$

and, hence,

$$|V(D_i)| \leq 4|E(D)|.$$

Therefore, $\varphi_i : \overline{D}_i \rightarrow D$ is an isomorphism for some i . \blacksquare

THEOREM 16.6.7 (Completeness). *Let $\langle X; R \rangle$ be a reduced finite presentation. If none of the probabilities in S is zero then B_S is \mathcal{L} -complete relative to the mapping $D \rightarrow \overline{D}$, i.e., $\overline{\Phi}_{B_S} = \mathcal{L}$.*

Proof. Let D be an arbitrary van Kampen diagram over $\langle X; R \rangle$. Since $\langle X; R \rangle$ is reduced it follows that D does not contain loops of length 1 (otherwise some of the generators in X would be trivial in $G = \langle X; R \rangle$) and each cell in D has vertex-simple boundary (otherwise a presentation $\langle X; R \rangle$ could be R -split). Now the result follows from Theorem 16.6.6. \blacksquare

16.6.3. Some properties of B_S . In this section we consider the random generator RG_χ defined in Section 16.5.4 relative to the basic extension $B = B_S$ defined in Section 16.6.1 and show that B_S satisfies properties (39), (40), and (41) from Section 16.5.4 and, therefore, functions X_n and K_n are random variables and the random generator RG_χ is correctly defined for B_S .

The starting diagram D_0 which is used throughout Section 16.6 contains no free edges or cells and, hence, equality (41) is satisfied.

Recall that an extended diagram D is a triple $(D, M(D), A(D))$ where $M(D)$ and $A(D)$ are sets of marked and active vertices of D , respectively.

LEMMA 16.6.8. *If $D_j = B_S(D_i)$ then*

$$(47) \quad (\chi(D_j) - \chi(D_i)) + (|M(D_j)| - |M(D_i)|) = 1.$$

Therefore, $0 \leq \chi(D_j) - \chi(D_i) \leq 1$ and condition (39) holds for B_S .

Proof. The basic extension B_S performs exactly one of the following actions. It either increases the geometric complexity χ of the input by 1 (when adds a cell or a free edge) or marks a vertex. Hence the result. \blacksquare

PROPOSITION 16.6.9. *Let L be the length of a longest relator in the given R -reduced presentation $\langle X; R \rangle$ and $D_0 = D_{m_0}, D_{m_1}, \dots, D_{m_k}$ be a sequence of marked diagrams such that $D_{m_{i+1}} = B_S(D_{m_i})$. Then $\chi(D_{m_k}) \geq \frac{k}{L+1}$ and the condition (40) holds for B_S .*

Proof. As proved in Lemma 16.6.8 $(\chi(D_{m_{i+1}}) - \chi(D_{m_i})) + (|M(D_{m_{i+1}})| - |M(D_{m_i})|) = 1$. Since $\chi(D_0) + |M(D_0)| = 0$ it follows that $\chi(D_{m_k}) + |M(D_{m_k})| = k$. Assume that $\chi(D_{m_k}) < \frac{k}{L+1}$. Then the number of vertices in D_{m_k} is not greater than $\frac{k}{L+1}L$. Therefore, the number of marked vertices $|M(D_{m_k})|$ is not greater than $\frac{k}{L+1}L$ and

$$\chi(D_{m_k}) + |M(D_{m_k})| < \frac{k}{L+1} + \frac{k}{L+1}L = k.$$

The obtained contradiction finishes the proof. \blacksquare

LEMMA 16.6.10. *If probabilities in S are non-zero then for each diagram $D \in \mathcal{K}$ there exists $D^* = B_S(D)$ such that $\chi(D^*) - \chi(D) = 1$, so the condition (43) holds for B_S .*

Proof. A diagram D^* is obtained from D by adding a cell or a free edge at the active vertex of D . \blacksquare

COROLLARY 16.6.11. *If probability in S are non-zero then B_S satisfies all properties (39)-(41) and (43). In particular, $\{Q_n\}_{n \in \mathbb{N}}$ and $\{\widehat{Q}_n\}_{n \in \mathbb{N}}$ are complete sets of termination conditions.*

COROLLARY 16.6.12. *The sets $\mathcal{K}_i = \{D \in \mathcal{K} \mid \chi(D) = i\}$ and $\mathcal{L}_i = \{D \in \mathcal{L} \mid \chi(D) = i\}$ form partitions of \mathcal{K} and \mathcal{L} , respectively. Therefore, the asymptotic densities ρ_χ on \mathcal{K} and \mathcal{L} (with respect to B_S) are well-defined.*

16.7. Asymptotic properties of diagrams

In this section we describe several asymptotic properties of diagrams relative to the basic extension B_S and, the sequence of termination conditions $\{\widehat{Q}_i\}_{i \in \mathbb{N}}$. In particular, we discuss asymptotic behavior of the length of the perimeter and the depth of diagrams relative to their size.

16.7.1. Properties related to RG_χ . Let $\langle X; R \rangle$ be a reduced finite presentation, \mathcal{K} the set of all marked diagrams, \mathcal{L} the set of all diagrams over $\langle X; R \rangle$, $\mathcal{K}_n = \{D \in \mathcal{K} \mid \chi(D) = n\}$, and $\mathcal{L}_n = \{\overline{D} \mid D \in \mathcal{K}_n\}$.

In Section 16.6.3 we introduced a discrete probability measure $P_{\mathcal{L}_n}$ on the set \mathcal{L}_n . The probability $P_{\mathcal{L}_n}$ depends on the basic extension operator B_S from Section 16.6.1. Below we freely use notation from Section 16.5 and 16.6.

If $D_j = B_S(D_k)$ then $\overline{D_j}$ can be obtained from $\overline{D_k}$ by either adding of a free edge or cell, or making a few foldings. Thus going along a trajectory λ the basic extension B_S adds some cells and free edges (we refer to them as *units*). Denote by $\mathcal{U}_i = \mathcal{U}_i(\lambda)$ the i th unit that was added by B_S along λ . Formally, one can express this as

$$\mathcal{U}_i = \mathcal{U}_i(\lambda) = D(\lambda_{X_i(\lambda)}) - D(\lambda_{X_i(\lambda)-1}),$$

where $X_i = \min\{j \mid \chi(D(\lambda_j)) = i\}$. Now we can introduce random variables $\xi_i : \Lambda \rightarrow \{0, 1\}$, $i \in \mathbb{N}$ defined on $\lambda \in \Lambda$ as follows:

$$\xi_i(\lambda) = \begin{cases} 0, & \text{if } \mathcal{U}_i(\lambda) \text{ in } D(\lambda_k) \text{ shares an edge with } \partial D(\lambda_k) \text{ for every } k \geq X_i(\lambda); \\ 1, & \text{otherwise.} \end{cases}$$

Notice that if $\xi_i(\lambda) = 0$ then $\mathcal{U}_i(\lambda)$ has depth one in $D(\lambda_k)$ for every $k \geq X_i(\lambda)$. Similarly, one can define ξ_i on $V(\mathcal{T})$. For $\theta \in V(\mathcal{T})$ with $\chi(D(\theta)) \geq i$ denote by

$\mathcal{U}_i(\theta)$ the i th unit that B_S added to $D(\theta)$ going along the path θ . Put

$$\xi_i(\theta) = \begin{cases} 0, & \chi(D(\theta)) < i \text{ or } \mathcal{U}_i(\theta) \text{ shares an edge with } \partial D(\theta); \\ 1, & \text{otherwise.} \end{cases}$$

Clearly, if $\lambda = (\lambda_0, \lambda_1, \lambda_2, \dots)$ then for any $s \leq t$ the following inequality holds:

$$(48) \quad \xi_i(\lambda_s) \leq \xi_i(\lambda_t) \leq \xi_i(\lambda).$$

Moreover, $\xi_i(\lambda_r) = \xi_i(\lambda)$ for some r . We denote the minimal such r by $r_i(\lambda)$.

LEMMA 16.7.1. *For any $i \in \mathbb{N}$ the function ξ_i is a random variable on Λ .*

Proof. Since each function ξ_i takes values from $\{0, 1\}$ it is enough to show that $\xi_i^{-1}(1) = \{\lambda \mid \xi_i(\lambda) = 1\}$ is measurable in Λ . Let $\lambda \in \xi_i^{-1}(1)$ and $r = r_i(\lambda)$. Then $\xi_i(\lambda_r) = 1$ and for any $\lambda' \in \text{Cone}(\lambda_r)$, $\xi_i(\lambda') = 1$. Thus,

$$\xi_i^{-1}(1) = \bigcup_{\lambda \in \xi_i^{-1}(1)} \text{Cone}(\lambda_{r(\lambda)})$$

is a countable union of cones, hence it is measurable, as claimed. \blacksquare

LEMMA 16.7.2. *Let $\theta \in V_{\widehat{Q}_n}$. Then for $i \leq n$,*

$$(49) \quad P_{\widehat{Q}_n}(\xi_i(\theta) = 1) < \frac{1}{2}.$$

Moreover, for any sequence (b_1, \dots, b_n) such that $b_1, \dots, b_n \in \{0, 1\}$,

$$(50) \quad P_{\widehat{Q}_n}(\xi_i(\theta) = 1 \mid \xi_j(\theta) = b_j \text{ where } j = 1, \dots, i-1, i+1, \dots, n) < \frac{1}{2}.$$

Proof. Let $\theta = e_1 e_2 \dots e_m \in V_{\widehat{Q}_n}$ and $\lambda = (\lambda_0, \lambda_1, \lambda_2, \dots) \in \text{Cone}(\theta)$. Notice that $\lambda_i = e_1 \dots e_i$, for $i = 0, \dots, m$. For $i = 0, \dots, m$ define

$$X_i(\theta) = \min\{j = 1, \dots, m \mid \chi(D(\lambda_j)) = i\}.$$

Clearly, $X_i(\theta)$ is the step at which the i th unit $\mathcal{U}_i = \mathcal{U}_i(\lambda)$ was added to the diagram. Let $D = D(\lambda_{X_i(\theta)})$ and v_i the active vertex of D , so \mathcal{U}_i is attached to v_i .

We define $Y_i(\theta)$ to be the least index $k \in \{1, \dots, m\}$ such that v_i is marked in $D(\lambda_k)$ and undefined otherwise. At the step $Y_i(\theta)$ the basic extension B_S folds some edges adjacent to v_i in $D(\lambda_{Y_i(\theta)})$. Notice that if Y_i is not defined on θ then $\xi_i(\theta) = 0$. Now, using a total probability formula:

$$\begin{aligned} P_{\widehat{Q}_n}(\xi_i(\theta) = 1) &= P_{\widehat{Q}_n}(\xi_i(\theta) = 1 \mid Y_i \text{ is defined on } \theta)P_{\widehat{Q}_n}(Y_i \text{ is defined on } \theta) \\ &\quad + P_{\widehat{Q}_n}(\xi_i(\theta) = 1 \mid Y_i \text{ is not defined on } \theta)P_{\widehat{Q}_n}(Y_i \text{ is not defined on } \theta) \\ &= P_{\widehat{Q}_n}(\xi_i(\theta) = 1 \mid Y_i \text{ is defined on } \theta)P_{\widehat{Q}_n}(Y_i \text{ is defined on } \theta) \\ &\leq P_{\widehat{Q}_n}(\xi_i(\theta) = 1 \mid Y_i \text{ is defined on } \theta). \end{aligned}$$

Therefore, we may assume in the statement of the lemma that Y_i is defined on θ . Denote by E_i the following event: one of the edges in $\mathcal{U}_i(\theta)$ adjacent to v_i belongs to $\partial D(\lambda_{Y_i(\theta)})$. Observe that E_i implies $\xi_i(\theta) = 0$. Clearly,

$$P_{\widehat{Q}_n}(\xi_i(\theta) = 1) = 1 - P_{\widehat{Q}_n}(\xi_i(\theta) = 0) \leq 1 - P_{\widehat{Q}_n}(E_i).$$

Now, by formula of total probability

$$\begin{aligned} P_{\widehat{Q}_n}(E_i) &= P_{\widehat{Q}_n}(E_i \mid \mathcal{U}_i \text{ is a cell})P_{\widehat{Q}_n}(\mathcal{U}_i \text{ is a cell}) \\ &\quad + P_{\widehat{Q}_n}(E_i \mid \mathcal{U}_i \text{ is a free edge})P_{\widehat{Q}_n}(\mathcal{U}_i \text{ is a free edge}). \end{aligned}$$

Observe that

$$P_{\widehat{Q}_n}(E_i \mid \mathcal{U}_i \text{ is a free edge}) = 1$$

and

$$P_{\widehat{Q}_n}(\mathcal{U}_i \text{ is a free edge}) + P_{\widehat{Q}_n}(\mathcal{U}_i \text{ is a cell}) = 1.$$

Thus

$$P_{\widehat{Q}_n}(E_i) > P_{\widehat{Q}_n}(E_i \mid \mathcal{U}_i \text{ is a cell})$$

unless $P_{\widehat{Q}_n}(E_i \mid \mathcal{U}_i \text{ is a cell}) = P_{\widehat{Q}_n}(E_i) = 1$.

Recall that B_s , when attaching a new cell to the current diagram D , chooses cells from R_{sym} uniformly and independently to the previous steps. We represent R_{sym} as a disjoint union of sets

$$R_{sym} = \bigcup_{x \in X \cup X^{-1}} R^{(x)},$$

where $R^{(x)}$ consists of all relators starting with x . It is easy to see (since $R_{sym}^{-1} = R_{sym}$) that for any presentation R and for any generator $x \in X \cup X^{-1}$,

$$(51) \quad \frac{|R^{(x)}|}{|R_{sym}|} \leq \frac{1}{2}.$$

Hence, for any fixed $x \in X^{\pm 1}$ and a uniformly chosen $r \in R_{sym}$ the probability that $x^{-1}r$ is not freely reduced is at most $\frac{1}{2}$. Thus

$$P_{\widehat{Q}_n}(E_i \mid \mathcal{U}_i \text{ is a cell}) \geq \frac{1}{2}.$$

Therefore, $P_{\widehat{Q}_n}(\xi_i = 1) \leq 1 - P_{\widehat{Q}_n}(E_i) < \frac{1}{2}$.

To see that (50) holds we observe that the argument above is valid for any choice of $\{b_i\}_{i \in \mathbb{N}}$. Indeed, the fold of the edges of \mathcal{U}_i adjacent to v_i depends only on the choice of a cell from R_{sym} which we attach to v_i and the edges on the boundary $\partial D(\lambda_{Y_i})$ adjacent to v_i . Since the cell \mathcal{U}_i is chosen uniformly and independently its choice does not affect the inequality (49). Notice also that the inequality (51) holds for any labels of the edges adjacent to v_i . ■

Define $S_n(\theta) = \sum_{i=1}^n \xi_i(\theta)$, for $\theta \in V(\mathcal{T})$.

LEMMA 16.7.3. *Let $\theta \in V_{\widehat{Q}_n}$. Then the length $l(D(\theta))$ of the perimeter of $D(\theta)$ satisfies the following inequality:*

$$l(D(\theta)) \geq n - S_n(\theta).$$

Proof. Observe that n is a total number of units in $D(\theta)$ and S_n is a total number of units that do not have edges on the boundary of $D(\theta)$. So $n - S_n(\theta)$ is a number of units that have at least one edge on the boundary, hence the result. ■

LEMMA 16.7.4. *Let $0 < \alpha < 1$. Then*

$$(52) \quad P_{\widehat{Q}_n} \left(S_n \geq \frac{(1+\alpha)}{2} n \right) \leq \exp \left(- \frac{3\alpha^2}{(12+4\alpha)} n \right).$$

Proof. Let Z_n be a binomial random variable with parameters $(n, \frac{1}{2})$. It follows from Lemma 16.7.2 that

$$P_{\widehat{Q}_n}(S_n \geq i) \leq p(Z_n \geq i).$$

By Chernoff inequality

$$p(Z_n \geq \mathbb{E}Z_n + t) \leq \exp\left(-\frac{t^2}{2(\mathbb{E}Z_n + t/3)}\right).$$

If $t = \alpha\mathbb{E}Z_n$, where $0 < \alpha < 1$, then

$$\begin{aligned} p(Z_n \geq (1 + \alpha)\mathbb{E}Z_n) &\leq \exp\left(-\frac{\alpha^2(\mathbb{E}Z_n)^2}{2(\mathbb{E}Z_n + \alpha\mathbb{E}Z_n/3)}\right) \\ &= \exp\left(-\frac{3\alpha^2}{(6+2\alpha)}\mathbb{E}Z_n\right), \end{aligned}$$

and, since $\mathbb{E}Z_n = \frac{n}{2}$, we get

$$P_{\widehat{Q}_n}\left(S_n \geq \frac{(1+\alpha)}{2}n\right) \leq p\left(Z_n \geq \frac{(1+\alpha)}{2}n\right) \leq \exp\left(-\frac{3\alpha^2}{(12+4\alpha)}n\right)$$

as required. \blacksquare

THEOREM 16.7.5. *Let $0 < \alpha < 1$. Then the following holds:*

- 1) Let $\mathcal{K}'_{n,\alpha} = \{D \in \mathcal{K}_n \mid l(D) < \frac{(1-\alpha)}{2}n\} \subseteq \mathcal{K}_n$. Then $P_{\mathcal{K}_n}(\mathcal{K}'_{n,\alpha}) \rightarrow 0$ exponentially fast as $n \rightarrow \infty$.
- 2) Let $\mathcal{L}'_{n,\alpha} = \{D \in \mathcal{L}_n \mid l(D) < \frac{(1-\alpha)}{2}n\} \subseteq \mathcal{L}_n$. Then $P_{\mathcal{L}_n}(\mathcal{L}'_{n,\alpha}) \rightarrow 0$ exponentially fast as $n \rightarrow \infty$.

Proof. Recall that

$$P_{\mathcal{K}_n}(D) = \sum_{\theta \in V_{\widehat{Q}_n}, D=D(\theta)} P_{\widehat{Q}_n}(\theta).$$

Clearly, $\theta \in V'_{\widehat{Q}_n}$ if and only if $D(\theta) \in \mathcal{K}'_n$ and, thus, $P_{\mathcal{K}_n}(\mathcal{K}'_{n,\alpha}) = P_{\widehat{Q}_n}(V'_{\widehat{Q}_n})$. So, it suffices to show that $P_{\widehat{Q}_n}(V'_{\widehat{Q}_n}) \rightarrow 0$ exponentially fast. By Lemma 16.7.3 $l(D(\theta)) \geq n - S_n(\theta)$. Hence

$$\begin{aligned} 0 \leq P_{\widehat{Q}_n}\left(l(D(\theta)) < \frac{(1-\alpha)}{2}n\right) &\leq P_{\widehat{Q}_n}(n - S_n(\theta) < \frac{(1-\alpha)}{2}n) \\ &= P_{\widehat{Q}_n}(S_n(\theta) \geq \frac{(1+\alpha)}{2}n). \end{aligned}$$

By (52) we have

$$P_{\widehat{Q}_n}\left(S_n(\theta) \geq \frac{(1+\alpha)}{2}n\right) \leq \exp\left(-\frac{3\alpha^2}{(12+4\alpha)}n\right) \rightarrow 0.$$

Hence the result. A similar argument proves 2). \blacksquare

COROLLARY 16.7.6. *Let $\langle X; R \rangle$ be an R -reduced presentation and \mathcal{L} be a set of representatives of all van Kampen diagrams over $\langle X; R \rangle$ (up to isomorphism). Put*

$$\mathcal{L}' = \{D \mid l(D) \geq \frac{1}{4}\chi(D)\}.$$

Then

$$\rho(\mathcal{L}') = \lim_{i \rightarrow \infty} P_{\mathcal{L}_i}(\mathcal{L}_i \cap \mathcal{L}') = 1.$$

Moreover, $P_{\mathcal{L}_i}(\mathcal{L}_i \cap \mathcal{L}') \rightarrow 1$ exponentially fast. Thus the set of all diagrams over $\langle X; R \rangle$ with linear isoperimetric function (with coefficient 4) is strongly generic with respect to the asymptotic density.

Let D be a diagram and d be the chain-distance metric on the set of free edges and cells. We define a ball $B_D(\mathcal{U}_i, r)$ in D with a center at the i th unit \mathcal{U}_i of radius r to be a subcomplex of D generated by $\{\mathcal{U}_j \mid d(\mathcal{U}_i, \mathcal{U}_j) \leq r\}$.

LEMMA 16.7.7. Let $\theta \in V_{\widehat{Q}_n}(\mathcal{T})$. If $\xi_i(\theta) = 0$ (where $i < n$) then $\mathcal{U}_i(\theta)$ has depth one in $D(\theta)$. Moreover, if there exists an index j such that $\mathcal{U}_j(\theta) \in B_{D(\theta)}(\mathcal{U}_i, r-1)$ and $\xi_j(\theta) = 0$ then the depth of $\mathcal{U}_i(\theta)$ in $D(\theta)$ is not greater than r .

Proof. Let $D = D(\theta)$. By definition of ξ_i if $\xi_i(\theta) = 0$ then \mathcal{U}_i shares an edge with ∂D and, hence, it has depth one in D .

Similarly, if $\mathcal{U}_j \in B_D(\mathcal{U}_i, r-1)$ and $\xi_j = 0$ then \mathcal{U}_j has depth one in D and by definition of $B_D(\mathcal{U}_i, r-1)$ the chain-distance between CU_i and CU_j is not greater than $r-1$. Hence the result. ■

PROPOSITION 16.7.8. Let δ be the depth function on diagrams. Then the expectation $\mathbb{E}\delta(D(\theta))$ of the function $\delta(D(\theta))$ on $(V_{\widehat{Q}_n}, P_{\widehat{Q}_n})$ is not greater than $\log n + 2$.

Proof. Let $\theta \in V_{\widehat{Q}_n}(\mathcal{T})$. For $i = 1, \dots, n$ define

$$d_i(\theta) = \max\{r \mid B(\mathcal{U}_i, r) \text{ does not contain } \mathcal{U}_j \text{ such that } \xi_j(\theta) = 0\} + 1.$$

Notice that if $r < n$ then $|B(\mathcal{U}_i, r)| \geq r+1$. Then from Lemma 16.7.2 it follows that

$$P_{\widehat{Q}_n}(d_i(\theta) \geq r) \leq \frac{1}{2^r}.$$

Define a random variable $d(\theta) = \max\{d_i(\theta) \mid i = 1, \dots, n\}$. From Lemma 16.7.7 and the definition of d_i it follows that

$$\delta(D(\theta)) \leq d(\theta)$$

and hence

$$\mathbb{E}\delta \leq \mathbb{E}d.$$

On the other hand,

$$(53) \quad \begin{aligned} P_{\widehat{Q}_n}(d(\theta) \geq r) &= P_{\widehat{Q}_n}(\bigvee_{i=1,\dots,n} (d_i(\theta) \geq r)) \\ &\leq \sum_{i=1,\dots,n} P_{\widehat{Q}_n}(d_i(\theta) \geq r) \leq \frac{n}{2^r}. \end{aligned}$$

Hence,

$$P_{\widehat{Q}_n}(d(\theta) - \log n \geq r) = P_{\widehat{Q}_n}(d(\theta) \geq r + \log n) \leq \frac{n}{2^{r+\log n}} = \frac{1}{2^r},$$

so

$$\mathbb{E}(d - \log n) \leq \sum_{r=1}^{\infty} \frac{r}{2^r} = 2.$$

Thus, $\mathbb{E}d \leq \log n + 2$, as claimed. ■

THEOREM 16.7.9. The following hold:

- 1) Let $\mathcal{K}_n'' = \{D \in \mathcal{K}_n \mid \delta(D) < 2 \log n\} \subseteq \mathcal{K}_n$. Then $P_{\mathcal{K}_n}(\mathcal{K}_n'') \rightarrow 1$ as $n \rightarrow \infty$.
- 2) Let $\mathcal{L}_n'' = \{D \in \mathcal{L}_n \mid \delta(D) < 2 \log n\} \subseteq \mathcal{L}_n$. Then $P_{\mathcal{L}_n}(\mathcal{L}_n'') \rightarrow 1$ as $n \rightarrow \infty$.

Proof. Let

$$V_{\widehat{Q}_n}'' = \{\theta \in V_{\widehat{Q}_n} \mid \delta(D(\theta)) \leq 2 \log n\}.$$

In Proposition 16.7.8 we defined a random variable $d(\theta)$ on $V_{\widehat{Q}_n}$ such that

$$d(\theta) \geq \delta(D(\theta)) \text{ and } P_{\widehat{Q}_n}(d(\theta) \geq r) \leq \frac{n}{2^r}.$$

This implies that

$$1 \geq P_{\hat{Q}_n}(V''_{\hat{Q}_n}) \geq P_{\hat{Q}_n}(d(D(\theta)) \leq 2 \log n) = 1 - P_{\hat{Q}_n}(d(D(\theta)) \geq 2 \log n) \geq 1 - \frac{1}{n} \rightarrow 1$$

as n tends to ∞ . Recall that

$$P_{\mathcal{K}_n}(D) = \sum_{\theta \in V_{\hat{Q}_n}, D=D(\theta)} P_{\hat{Q}_n}(\theta).$$

Clearly, $\theta \in V''_{\hat{Q}_n}$ if and only if $D(\theta) \in \mathcal{K}_n''$. Thus,

$$P_{\mathcal{K}_n}(\mathcal{K}_n'') = P_{\hat{Q}_n}(V''_{\hat{Q}_n}) \rightarrow 1$$

as n tends to infinity.

A similar argument proves 2). ■

THEOREM 16.7.10. *Let $\langle X; R \rangle$ be an R -reduced presentation and $0 < \alpha < 1$. The following holds:*

- 1) Let $\mathcal{K}_{n,\alpha}''' = \{D \in \mathcal{K}_n \mid \delta(D) < 2 \log n \text{ \& } l(D) \geq \frac{(1-\alpha)}{2}n\} \subseteq \mathcal{K}_n$. Then $P_{\mathcal{K}_n}(\mathcal{K}_{n,\alpha}''') \rightarrow 1$ as $n \rightarrow \infty$.
- 2) Let $\mathcal{L}_{n,\alpha}''' = \{D \in \mathcal{L}_n \mid \delta(D) < 2 \log n \text{ \& } l(D) \geq \frac{(1-\alpha)}{2}n\} \subseteq \mathcal{L}_n$. Then $P_{\mathcal{L}_n}(\mathcal{L}_{n,\alpha}''') \rightarrow 1$ as $n \rightarrow \infty$.

Proof. Follows from equalities $\mathcal{K}_{n,\alpha}''' = \mathcal{K}'_{n,\alpha} \cap \mathcal{K}_n''$ and $\mathcal{L}_{n,\alpha}''' = \mathcal{L}'_{n,\alpha} \cap \mathcal{L}_n''$ and Theorems 16.7.5 and 16.7.9. ■

16.8. Generic properties of trivial words

In this section we investigate properties of random trivial words over $\langle X; R \rangle$.

16.8.1. Random trivial words. Denote by $WP(X; R)$ the set of all cyclic words representing the identity of the group $G = \langle X; R \rangle$. In this section we define a discrete probability measure on $WP(X; R)$.

Recall that \mathcal{L}_n is a set of all van Kampen diagrams over $\langle X; R \rangle$ of a size n and $P_{\mathcal{L}_n}$ is a discrete probability measure on \mathcal{L}_n . Denote by CW_i (for $i \in \mathbb{N}$) the set of boundary labels (as cyclic words) of diagrams from \mathcal{L}_i and by \overline{CW}_n the union

$$\overline{CW}_n = \bigcup_{i=1}^n CW_i.$$

It follows from van Kampen Lemma that

$$WP(X; R) = \bigcup_{i=1}^{\infty} CW_i = \bigcup_{i=1}^{\infty} \overline{CW}_i.$$

Clearly,

$$\overline{CW}_1 \subseteq \overline{CW}_2 \subseteq \overline{CW}_3 \subseteq \dots$$

One can induce probability measures from $(\mathcal{L}_n, P_{\mathcal{L}_n})$ onto the sets CW_n and \overline{CW}_n as follows. For $S \subseteq CW_n$ and $S' \subseteq \overline{CW}_n$ put

$$P_{CW_n}(S) = P_{\mathcal{L}_n}(\{D \in \mathcal{L}_n \mid \text{the boundary label of } D \text{ belongs to } S\}),$$

$$P_{\overline{CW}_n}(S') = \frac{\sum_{i=1}^n P_{CW_i}(S' \cap CW_i)}{n}.$$

It is easy to check that P_{CW_n} and $P_{\overline{CW}_n}$ are discrete probability measures on CW_n and \overline{CW}_n .

One can describe the probability measure P_{CW_n} in terms of the following random generator.

ALGORITHM 16.8.1 (Random Generator I of trivial words).

INPUT. A number $n \in \mathbb{N}$.

OUTPUT. A word w such that $w =_G 1$.

COMPUTATIONS.

- 1) Run RG_χ to generate a random diagram D of size n .
- 2) Output the boundary label of D .

The probability measure $P_{\overline{CW}_n}$ can be described in terms of the following random generator.

ALGORITHM 16.8.2 (Random Generator II of trivial words).

INPUT. A number $n \in \mathbb{N}$.

OUTPUT. A word w such that $w =_G 1$.

COMPUTATIONS.

- 1) Generate randomly and uniformly a number i from the set $\{1, \dots, n\}$.
- 2) Run RG_χ to generate a random diagram D of size i .
- 3) Output the boundary label of D .

In view of the random generators above the probability measures P_{CW_n} and $P_{\overline{CW}_n}$ are very natural.

16.8.2. Generic properties of trivial words. In this section we study generic properties of words representing the trivial element of $G = \langle X; R \rangle$.

Fix α such that $0 < \alpha < 1$ and define

$$\overline{CW}_{n,\alpha} = \{w \in \overline{CW}_n \mid w \text{ is a boundary label of some } D \in \bigcup_{i=1}^n \mathcal{L}_{i,\alpha}'''\}.$$

THEOREM 16.8.3. *Let $G = \langle X; R \rangle$. The following holds:*

$$P_{\overline{CW}_n}(\overline{CW}_{n,\alpha}) \rightarrow 1 \text{ as } n \rightarrow \infty.$$

Proof. By definition of $P_{\overline{CW}_n}$ and $\overline{CW}_{n,\alpha}$ we have

$$\begin{aligned} P_{\overline{CW}_n}(\overline{CW}_{n,\alpha}) &= \frac{1}{n} \sum_{i=1}^n P_{CW_i}(\overline{CW}_{n,\alpha} \cap CW_i) \\ &= \frac{1}{n} \sum_{i=1}^n P_{CW_i}(\{w \in CW_i \mid w \text{ is a boundary label of some } D \in \mathcal{L}_{i,\alpha}'''\}) \\ (54) \qquad \qquad \qquad &= \frac{1}{n} \sum_{i=1}^n P_{\mathcal{L}_i}(\mathcal{L}_{i,\alpha}'''). \end{aligned}$$

Now, if we denote $\sum_{i=1}^n P_{\mathcal{L}_i}(\mathcal{L}_{i,\alpha}''')$ by a_n and put $b_n = n$ then (54) becomes equal to $\frac{a_n}{b_n}$. Notice that

$$\frac{a_{n+1} - a_n}{b_{n+1} - b_n} = P_{\mathcal{L}_i}(\mathcal{L}_{i,\alpha}''').$$

Since $b_n = n$ is strictly increasing with n and tending to infinity, and, as proved in Theorem 16.7.10, $\lim_{i \rightarrow \infty} P_{\mathcal{L}_i}(\mathcal{L}_{i,\alpha}''') = 1$ we can apply the Stolz-Cesaro theorem and obtain

$$\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = 1$$

and, hence, $P_{\overline{CW}_n}(\overline{CW}_{n,\alpha}) \rightarrow 1$ as $n \rightarrow \infty$. ■

Let

$$CW^{(\alpha)} = \bigcup_{i=0}^{\infty} \overline{CW}_{i,\alpha}.$$

THEOREM 16.8.4. *Let $\langle X; R \rangle$ be a finite symmetrized reduced presentation and $G = \langle X; R \rangle$. Then the following holds.*

- 1) *The time complexity function for Algorithm \mathcal{A} (the decision algorithm for the word problem in G) on the set of inputs $w \in CW^{(\alpha)}$ is bounded from above by the polynomial*

$$O(|w|^{2+2 \log L(R)}).$$

- 2) *The time complexity function for Algorithm \mathcal{B} (the algorithm for the search word problem in G) on the set of inputs $w \in CW^{(\alpha)}$ is bounded by the polynomial*

$$O(|w|^{4+4 \log L(R)}).$$

Proof. Denote $\frac{2}{1-\alpha}$ by β . Let w be an arbitrary word in $CW^{(\alpha)}$. Then w is a boundary label of some $D \in \bigcup_{i=1}^{\infty} \mathcal{L}_{i,\alpha}'''$. Assume that $D \in \mathcal{L}_{n,\alpha}'''$ for some $n \in \mathbb{N}$. It follows from the definition of the sets $\mathcal{L}_{i,\alpha}'''$ that $n < \frac{2|w|}{1-\alpha} = \beta|w|$ and

$$(55) \quad \delta(D) < 2 \log n < 2 \log \frac{2|w|}{1-\alpha} < 2 \log \beta + 2 \log |w|.$$

By Theorem 16.4.3 the number of steps required for Algorithm \mathcal{A} to terminate on the input w is bounded from above by

$$O(\delta(w)|w| \cdot L(R)^{\delta(w)} \log(|w|L(R))),$$

where $\delta(w)$ is a depth of w and $L(R)$ is a total length of R . It follows that $\delta(w) \leq \delta(D) < 2 \log \beta + 2 \log |w|$. Now the formula above becomes

$$\begin{aligned} & O(2(\log \beta + \log |w|)|w| \cdot L(R)^{2 \log \beta + 2 \log |w|} \log(|w|L(R))) \\ & = O(|w|^2 L(R)^{2 \log |w|}) = O(|w|^{2+2 \log L(R)}). \end{aligned}$$

This proves 1).

By Theorem 16.4.10 the number of steps required for Algorithm \mathcal{B} to terminate on the input w is bounded from above by

$$O(|w|^2 L(R)^{2\delta(w)} (2|w| + \delta(w)M(R))^2),$$

where $M(R) = \max\{|r| \mid r \in R\}$. From (55) we deduce that

$$\begin{aligned} & O(|w|^2 L(R)^{4 \log \beta + 4 \log |w|} (2|w| + (2 \log \beta + 2 \log |w|)M(R))^2) \\ & = O(|w|^4 L(R)^{4 \log |w|}) = O(|w|^{4+4 \log L(R)}). \end{aligned}$$
■

16.9. Comparison with standard techniques

In this section we briefly compare Algorithm \mathcal{A} with the general techniques for the word search problem such as Todd-Coxeter procedure and the total enumeration of $gp_F(R)$. The Knuth-Bendix-like procedures are harder to compare since they dynamically change the presentation of a group, so we omit them.

16.9.1. The Todd-Coxeter algorithm. Let $\langle X; R \rangle$ be a finite symmetrized presentation and $G = \langle X; R \rangle$. Let $\Gamma_0 = \Gamma(\varepsilon)$ be an X -digraph with one vertex (which is a base vertex) denoted by v_0 and no edges. The Todd-Coxeter algorithm works the following way. It starts with Γ_0 and applies R -extensions until there is a loop starting at the base point v_0 labelled with w . Below we are looking for an upperbound for the worst case time complexity for Todd-Coxeter algorithm.

Let D be a van Kampen diagram, v_0 a vertex in ∂D , and w a boundary label of D which is read starting at v_0 in a counterclockwise direction. Denote by $\bar{\delta}(D)$ the depth $\delta_{v_0}(D)$ of D with respect to the vertex v_0 and by $\bar{\delta}(w)$ the minimum among all such diagrams D .

The next proposition is analogous to Proposition 16.3.14.

PROPOSITION 16.9.1. *Let D be a diagram with a boundary label w , $m = \delta(D)$, and $\varphi : \Gamma(w) \rightarrow \partial D$ be a morphism. Then there exists a morphism of 2-complexes $\psi : D \rightarrow \mathcal{C}^{(m)}(\Gamma_0)$ such that the following diagram commutes.*

$$\begin{array}{ccc} \Gamma_0 & \xrightarrow{\varphi} & D \\ & \searrow \varphi_C & \downarrow \psi \\ & & \mathcal{C}^{(m)}(\Gamma_0) \end{array}$$

Proof. One can prove the assertion of the proposition in a fashion similar to the proof of Proposition 16.3.14. ■

COROLLARY 16.9.2. *Let $\langle X; R \rangle$ be a finite symmetrized presentation, $G = \langle X; R \rangle$, and $w \in gp_F(R)$. The time complexity of Todd-Coxeter procedure is bounded by $O(L(R)^{\bar{\delta}})$.*

The fraction of uppebounds of time-complexities of Algorithm \mathcal{A} and the Todd-Coxeter procedure can be roughly estimated by

$$|w|L(R)^{\delta - \bar{\delta}}.$$

Clearly, for each word $w \in gp_F(R)$ we have $\bar{\delta}(w) \geq \delta(w)$, but it is hard to say how large (generically or on average) the difference $\bar{\delta}(w) - \delta(w)$ is. Note that it is easy to construct a series of words for which Todd-Coxeter has exponential time complexity and Algorithm \mathcal{A} is linear. For instance, this happens for $G = \langle a, b; [a, b] \rangle$ and $w_i = (ab)^i(a^{-1}b^{-1})^i$. Since G is aspheric the reduced diagram for w_i is unique and it is easy to see that it consists of i diagonal blocks in the first quarter of the grid. Therefore, $\bar{\delta}(w_i) = i$ and $\delta(w_i) = 1$.

We performed a series of experiments for different classes of groups (mostly one-relator groups) and obtained the following results. For most of the randomly generated words $w \in gp_F(R)$ $\delta(w) = 1$ and $\delta(\bar{w}) = \log |w|$.

16.9.2. Total enumeration of $gp_F(R)$. Let $G = \langle X; R \rangle$ and $w = w(X)$. For $k \in \mathbb{N}$ define

$$C_k = \left\{ \prod_{i=1}^m c_i^{-1} r_{k_i} c_i \mid m \leq k, |c_i| \leq k \right\}.$$

Clearly, the sets C_r are finite and $gp_F(R) = \bigcup_{k \in \mathbb{N}} C_k$. Total enumeration of $gp_F(R)$ enumerates words from C_0 , C_1 and so on, until it finds w . To enumerate C_k one has to enumerate up to k conjugates of length up to k , which has the total time complexity $O(3^{k^2})$ (we do not consider a case of a cyclic group G). Therefore, the complexity of the enumeration of $gp_F(R)$ is bounded from below by $O(3^{\widehat{\delta}^2})$, where $\widehat{\delta} = \widehat{\delta}(w)$ is the least number k such that $w \in C_k$. It is hard to estimate the value $\widehat{\delta}(w)$ in terms of $|w|$. Though the following proposition holds.

PROPOSITION 16.9.3. *Let $\langle X; R \rangle$ be a finite symmetrized presentation, $G = \langle X; R \rangle$, and $w \in gp_F(R)$. Then $\widehat{\delta}(w) \geq \overline{\delta}(w)$.*

Proof. Let k be the smallest number such that

$$w = \prod_{i=1}^m c_i^{-1} r_{k_i} c_i$$

where $m \leq k$, $|c_i| \leq k$. Let D_1 be a diagram which is a bouquet of diagrams corresponding to $c_i^{-1} r_{k_i} c_i$ and D_2 is a diagram obtained from D_1 by foldings of the boundary ∂D_1 (so w is a boundary label of D_2). Since Stallings' folds preserve the incidence in X -digraph it follows that the chain distance from the initial vertex of D_2 to any cell is less or equal than k . This proves the proposition. ■

The obvious corollary of the proposition above is that the total enumeration has much worse time complexity than that of the standard Todd-Coxeter procedure.

CHAPTER 17

Conjugacy Search Problem

17.1. Introduction

The conjugacy problem is the second problem in the list of the fundamental problems. As well as the word problem, the conjugacy problem is unsolvable in general [214] (also see [199] and [200]). Clearly, if the word problem is unsolvable then the conjugacy problem is unsolvable too. Moreover, for almost all classes with the solvable conjugacy problem, its actual complexity is higher than the complexity of the word problem. So, we can say that the conjugacy problem is harder than the word problem.

We briefly list some known positive results about the conjugacy problem in groups. In abelian groups the conjugacy problem is equivalent to the word problem and, hence, is solvable by Gauss elimination procedure. For small-cancellation groups it was shown by Greendlinger in [106] that the variation of the Dehn's algorithm solves the conjugacy problem. For word-hyperbolic it was shown by Holt that the conjugacy problem is solvable in quadratic time in terms of the lengths of the given words. It is an open problem for automatic groups whether the conjugacy problem is solvable or not, though all interesting automatic groups are known to be biautomatic and, hence have the conjugacy problem solvable in quadratic time. It is solvable for braid groups, though there is no good upperbound on the time-complexity of the algorithm. It is also solvable for nilpotent and polycyclic groups. It is an open question whether it is solvable for $\text{Aut}(F_n)$.

In this part of the chapter we study the generic-case complexity of the conjugacy search problem. We design an algorithm (Algorithm \mathcal{A}_C , 17.6.11) for solving it and show that for almost all presentations (including all known to us examples of groups with undecidable conjugacy problem) the conjugacy search problem is generically polynomial in terms of the lengths of words, where the degree depends on the presentation. Presentations non-covered by our proofs can be considered similarly. Another good part of our algorithm is that it does not require any additional information about a presentation or elements (like a constant of hyperbolicity for hyperbolic groups or existence of a good epimorphism onto an infinite hyperbolic group). One can start the algorithm on any presentation and a pair of words. No precomputation required.

The idea of the proposed algorithm is (somewhat) similar to the well-known Todd-Coxeter algorithm for the word search problem. We show the existence of some universal construction (we call it the conjugacy graph, compare to the Cayley graph) which reflects the structure if the conjugacy class of the given word. So, the algorithm constructs two conjugacy graphs and if at some step they contain certain loops then algorithm stops with a positive answer. The algorithm does not have the negative answer. If non-conjugate words are plugged in then it never stops.

The proposed algorithm, also, allows one to determine many other numerical characteristics of elements of the given group. For example, if a word w is of finite order then it will find it eventually. Of course, the same can be done by a coset enumeration, but we claim that our algorithm is more efficient. If w_1 and w_2 are two words in generators of $G = \langle X; R \rangle$ such that

$$w_1^{n_1} \sim_G w_2^{n_2}$$

then the algorithm will eventually find (describe) the whole set of pairs with that property.

Finally, we would like to point out that this algorithm is one of its kind. There are algorithms working for particular classes of groups, but none of them work in full generality.

17.2. Weighted graphs

In this section we define main objects of study, namely conjugacy graphs and pseudo conjugacy graphs, that are analogous to Schreier graphs of a subgroups of a finitely presented group.

17.2.1. Definition. We start from the definition of X -digraphs. Let X be a finite alphabet closed under inversions (i.e., $X = X^{-1}$) and $\Gamma = (V, E)$ be a directed graph with an edge labelling function $\mu : E \rightarrow X$. For each edge $e = u_1 \rightarrow u_2 \in E$ we will assume the existence of the *inverse edge* $e^{-1} = u_2 \rightarrow u_1$ labeled with a symbol $\mu(e^{-1}) = \mu(e)^{-1}$ without actually adding it into E . A pair (Γ, μ) is called an *X -digraph*. Often we will refer to (Γ, μ) as to Γ without specifying μ . For more information on X -digraphs see [145].

It will be convenient to use the following notation for X -digraphs. Let Γ be an X -digraph. By $V(\Gamma)$ and $E(\Gamma)$ we denote the set of vertices and edges in Γ respectively. If $e = u \rightarrow v \in E(\Gamma)$ then the *origin* u of e will be denoted by $\alpha(e)$ and the *terminus* v by $\beta(e)$. A *path* p in Γ is a sequence of edges $e_1 \dots e_k$ such that $\beta(e_i) = \alpha(e_{i+1})$ for each $1 \leq i \leq k-1$. An origin $\alpha(p)$ of a path p is an origin of its first edge and the terminus $\beta(p)$ of p is the terminus of its last edge. The origin and the terminus of an empty path can be any vertex of Γ , and, hence, are not defined. When we will talk about a concatenation of paths p_1 and p_2 we will assume that $\beta(p_1) = \alpha(p_2)$. One can extend function μ from the edge set E to the set of paths in a natural way. If $p = d_1 \dots d_k$ is a path in Γ then

$$\mu(p) = \mu(d_1) \dots \mu(d_k).$$

Now, we can define a weighted X -digraph. Let Γ be a connected X -digraph and γ a function $\gamma : E \rightarrow \mathbb{Z}$. If

$$\gamma(e) = -\gamma(e^{-1})$$

holds for any edge $e \in E(\Gamma)$ then γ is called a *weight function* on Γ and a pair (Γ, γ) is called a *weighted X -digraph*. Usually we shorten the notation (Γ, γ) omitting γ . In pictures we will depict each edge e of a weighted X -digraph Γ with a pair $(\mu(e), \gamma(e)) \in X \times \mathbb{Z}$ - the label and weight of e .

We extend function γ from the edge set E to the set of paths the following way. If $\pi = d_1 \dots d_k$ is a path in Γ then

$$\gamma(\pi) = \sum_{i=1}^k \gamma(d_i).$$

We say that a loop π in a weighted X -digraph (Γ, γ) is a *base loop* if $\gamma(\pi) = 1$.

A path $\pi = e_1 \dots e_k$ in Γ is *non-reduced* if $e_{i+1} = e_i^{-1}$ for some $1 \leq i \leq k-1$. Otherwise π is called *reduced*. To reduce a non-reduced path c means to remove all such pairs from π . Since removing $e_i e_{i+1}$ from c decreases the length of π by 2 in a finite number of steps a reduced path will be obtained. One can show that the final result of reductions does not depend on the sequence of reductions. Let π' be a reduced path obtained from π . Clearly $\gamma(\pi) = \gamma(\pi')$ and $\mu(\pi) =_{F(X)} \mu(\pi')$.

Similarly, π is *non-cyclically reduced* if it is not reduced or $e_1 = e_k^{-1}$. Otherwise it is called *cyclically reduced*. From this definition follows that non cyclically reduced c must be a loop. To cyclically reduce a non cyclically reduced path π we first freely reduce π and then remove first and last edges of the result while they are opposite. The result of such operation is clearly cyclically reduced and is uniquely defined. By induction on the number of single reductions it is easy to see that if π' is a cyclically reduced path obtained from π by a procedure defined above then $\gamma(\pi) = \gamma(\pi')$ and $\mu(\pi) \sim_{F(X)} \mu(\pi')$.

17.2.2. Conjugacy and pseudo conjugacy graphs. Let G be a group, $X \subseteq G$ a generating set for G such that $X^{-1} = X$, and $w = w(X)$. In this section we define conjugacy and pseudo conjugacy graphs of w in G (relative to X) and show that for any w in generators of G there exists at least one pseudo conjugacy graph. The existence of conjugacy graphs will be shown in Section 17.4.

Let Γ be a weighted X -digraph (not necessarily connected), w a word in generators $X \cup X^{-1}$, and $N \in \mathbb{N} \cup \{\infty\}$ (here ∞ is a symbol denoting the element which greater than any natural number).

DEFINITION 17.2.1. We say that a pair (Γ, N) is a *pseudo conjugacy graph* of the word w in G if the following conditions are satisfied:

(PCG1) For any loop π in Γ

$$(56) \quad \mu(\pi) \sim_G w^{\gamma(\pi)}.$$

(PCG2) Either $N = \infty$, or $N < \infty$ and $w^N =_G 1$.

Since cyclic reduction of a loop in a weighted X -digraph does not change its weight and label (as a cyclic word) we can slightly modify the first requirement (PCG1) in the definition of pseudo conjugacy graphs to the following one:

(PCG1)' For any cyclically reduced loop π in Γ $\mu(\pi) \sim_G w^{\gamma(\pi)}$.

The obtained definition is equivalent to the previous. Also, notice that if $N < \infty$ then for any loop π in Γ we have $\mu(\pi)^N =_G 1$.

DEFINITION 17.2.2. A connected folded pseudo conjugacy graph (Γ, N) of $w \in G$ is called a conjugacy graph of w in G (relative to the generating set X) if the following conditions hold:

(CG1) For every $v \in F(X)$ and $\gamma \in \mathbb{N}$ word v and w^γ conjugate if and only if there exists a loop π in Γ such that $\mu(\pi) = v$ and $\gamma(\pi) \equiv \gamma \pmod{N}$.

(CG2) N is an order of w in G .

Let $w = w_1 \dots w_k$ be a word in generators $X \cup X^{-1}$ of the group G and Γ is the graph $Loop_1(w)$ depicted in Figure 17.1). Then the pair (Γ, ∞) is a pseudo conjugacy graph of w in G . The graph Γ consists of k vertices $\{1, \dots, k\}$ and k edges $e_i = i \xrightarrow{(w_i, 0)} i+1$ ($i = 1, \dots, k-1$) and an edge $e_k = k \xrightarrow{(w_k, 1)} 1$. It is

straightforward to check that $\text{Loop}_1(w)$ is, indeed, a pseudo conjugacy graph of w .

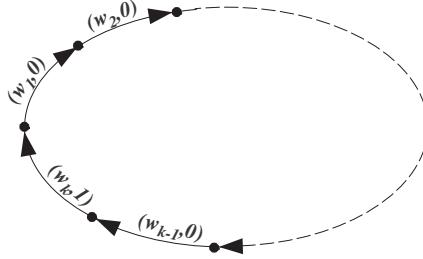


FIGURE 17.1. Graph $\Gamma(w)$.

LEMMA 17.2.3 (Conjugators in pseudo conjugacy graphs). *Let Γ be a pseudo conjugacy graph of w . Suppose that π_0 and π_1 are loops in Γ such that $\gamma(\pi_0) = 1$ and $p = d_1 \dots d_k$ be a path starting at the initial vertex of π_0 (hence $\alpha(p) = \alpha(\pi_0)$) and terminating at the initial vertex of π_1 (hence $\beta(p) = \alpha(\pi_1)$). Then $\mu(\pi_1) =_G \mu(p)^{-1} \mu(\pi_0)^{\gamma(\pi_1)} \mu(p)$.*

Proof. From the assumption put on the path p the sequence of edges $p^{-1} \pi_0^{\gamma(\pi_1)} p \pi_1^{-1}$ is a loop in Γ (perhaps not cyclically reduced). From assumptions put on π_0 and π_1 we have $\gamma(p^{-1} \pi_0^{\gamma(\pi_1)} p \pi_1^{-1}) = -\gamma(p) + \gamma(\pi_1)\gamma(\pi_0) + \gamma(p) - \gamma(\pi_1) = 0$. Since Γ is a pseudo conjugacy graph we have $\mu(p^{-1} \pi_0^{\gamma(\pi_1)} p \pi_1^{-1}) =_G 1$. Thus $\mu(p)^{-1} \mu(\pi_0)^{\gamma(\pi_1)} \mu(p) \mu(\pi_1)^{-1} =_G 1$. ■

The following assertion is a corollary of Lemma 17.2.3.

PROPOSITION 17.2.4. *Let G be a group, X be the generating set for G closed under inversions, (Γ, N) a pseudo conjugacy graph of w in G , and π a base loop in (Γ, N) . Then (Γ, N) is a pseudo conjugacy graph of $\mu(\pi)$ in G .*

The existence of a base loop in a pseudo conjugacy graph is important. Further in the sequel when we say that Γ is a pseudo conjugacy graph of w in G we assume that, in addition to properties (PCG1) and (PCG2), there exists a base loop in Γ labeled with w .

Let (Γ_1, N_1) and (Γ_2, N_2) be pseudo conjugacy graphs of w in G , and $\varphi : \Gamma_1 \rightarrow \Gamma_2$ an X -digraph morphism. We say that φ is a *pseudo conjugacy graph morphism* or, to shorten, *PCG-morphism* if the following conditions hold:

- (M1) there exists $c \in \mathbb{N} \cup \{\infty\}$ such that $N_1 = cN_2$;
- (M2) for any loop l in Γ_1 $\gamma(l) \equiv \gamma(\varphi(l)) \pmod{N_2}$.

17.3. Transformations of weighted X -digraphs

In this section we define several transformations of weighted X -digraphs. All together they will be used to construct conjugacy graphs and to solve the conjugacy search problem for finitely presented groups. Let $\langle X; R \rangle$ be a finite group presentation, $w = w(X)$, and Γ a weighted X -digraph. Each of the transformations will be shown to satisfy the following important property. The result of a transformation

of Γ is a pseudo conjugacy graph of w in G if and only if Γ is a pseudo conjugacy graph of w in G .

17.3.1. Shift operator. Shift operator is a transformation of a weighted X -digraph Γ which preserves its X -digraph structure and changes the weights of edges incident to some vertex in Γ .

DEFINITION 17.3.1. Let v be a vertex of a weighted X -digraph Γ and $\varepsilon \in \mathbb{Z} \setminus \{0\}$. A shift of the weight at the vertex v in Γ by ε is the following transformation of Γ :

- 1) Each edge $v \xrightarrow{(x,\gamma)} u$ (where $v \neq u$) is replaced with an edge $v \xrightarrow{(x,\gamma-\varepsilon)} u$.
- 2) Each edge $u \xrightarrow{(x,\gamma)} v$ (where $v \neq u$) is replaced with an edge $u \xrightarrow{(x,\gamma+\varepsilon)} v$.

(Observe, that we do not replace edges $v \xrightarrow{(x,\gamma)} v$.) The result of a shift operator is denoted by $Shift_\varepsilon(\Gamma, v)$. See Figure 17.2.

Since $Shift_\varepsilon$ changes only a weight function γ on an X -digraph Γ there exists a natural X -digraph isomorphism

$$\varphi_{Shift} : \Gamma \rightarrow Shift_\varepsilon(\Gamma, v).$$

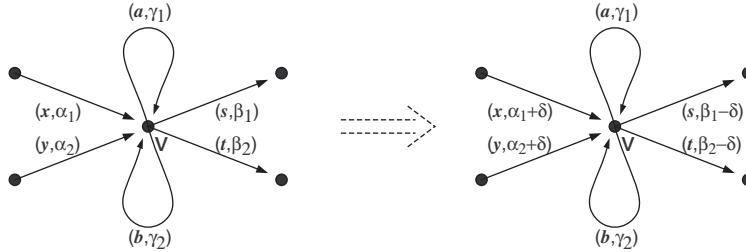


FIGURE 17.2. Shift operator.

PROPOSITION 17.3.2. Let G be a group, X a generating set for G closed under inversions, Γ a weighted X -digraph, $v \in V(\Gamma)$, $\varepsilon \in \mathbb{Z}$, and $N \in \mathbb{N}$. Then the following hold:

- (Sh1) A pair (Γ, N) is a pseudo conjugacy graph of $w \in G$ if and only if $(Shift_\varepsilon(\Gamma, v), N)$ is.
- (Sh2) If a pair (Γ, N) is a pseudo conjugacy graph of $w \in G$ then φ_{Shift} is a PCG-morphism.

Proof. Denote $Shift_\varepsilon(\Gamma, v)$ by Γ' . Let $\varphi_{Shift} : \Gamma \rightarrow \Gamma'$ be a corresponding X -digraph isomorphism, $\pi = e_1 \dots e_k$ an arbitrary loop in Γ and

$$\pi' = \varphi_{Shift}(\pi) = \varphi_{Shift}(e_1) \dots \varphi_{Shift}(e_k)$$

an image of π in Γ' . Clearly $\mu(\pi) = \mu(\pi')$. To finish the proof of (Sh1) and (Sh2) it is enough to show that $\gamma(\pi) = \gamma(\pi')$.

Let v_1, \dots, v_{k+1} ($v_1 = v_{k+1}$) be a sequence of vertices which π passes through in Γ . If for each $i = 1, \dots, k$ $v_i \neq v$ then for each $i = 1, \dots, k$ $\gamma(e_i) = \gamma(\varphi_{Shift}(e_i))$ and hence $\gamma(\pi) = \gamma(\pi')$ due to additivity of γ . If for each $i = 1, \dots, k$ $v = v_i$

then each $e_i = (v, v)$ and, therefore, for each $i = 1, \dots, k$ $\gamma(e_i) = \gamma(\varphi_{Shift}(e_i))$ and $\gamma(\pi) = \gamma(\pi')$.

Now let $v = v_i$ for some $i = 1, \dots, k$ but not for all of them. We may assume that $v \neq v_1$ (take a cyclic permutation of π if required). Let $1 < s, t < k+1$ be such that $v = v_j$ for all $j = s, \dots, t$ and $v \neq v_{s-1}$ and $v \neq v_{t+1}$. Then $e_{s-1} = (u, v)$ (for some $u' \neq v$), $e_j = (v, v)$ ($j = s, \dots, t-1$) and $e_t = (v, u')$ (for some $u' \neq v$). Using the definition of $Shift_{(v, \varepsilon)}$ we get $\gamma(e_{s-1} \dots e_t) = \gamma(\varphi_{Shift}(e_{s-1} \dots e_t))$. Finally, notice that different such subpaths do not overlap in π and weights of edges not belonging to them do not change. Thus $\gamma(\pi) = \gamma(\pi')$. ■

PROPOSITION 17.3.3. *Let $\tau : K_1 \rightarrow K_2$ be a PCG-morphism, $K'_1 = Shift_{\varepsilon_1}(K_1, v_1)$, and $K'_2 = Shift_{\varepsilon_2}(K_2, v_2)$. Then there exists a PCG-morphism $\theta : K'_1 \rightarrow K'_2$ such that the diagram commutes:*

$$\begin{array}{ccc} (K_1, N_1) & \xrightarrow{\tau} & (K_2, N_2) \\ \downarrow \varphi_{Shift}^{(1)} & & \downarrow \varphi_{Shift}^{(2)} \\ (K'_1, N_1) & \xrightarrow{\theta} & (K'_2, N_2) \end{array}$$

Proof. Since $\varphi_{Shift}^{(1)}$ and $\varphi_{Shift}^{(2)}$ are X -digraph isomorphisms it is natural to define θ equal to τ on the X -digraph level. To show that so defined X -digraph morphism θ is a PCG-morphism it is enough to prove the property (M2) for θ .

Let π'_1 be an arbitrary loop in K'_1 , $\pi_1 \in K_1$ its preimage, $\pi_2 = \tau(\pi_1)$, and $\pi'_2 = \varphi_{Shift}^{(2)}(\pi_2)$. Then $\theta(\pi'_1) = \pi'_2$. By Proposition 17.3.2 we have $\gamma(\pi_1) = \gamma(\pi'_1)$ and $\gamma(\pi_2) = \gamma(\pi'_2)$. Since τ is a PCG-morphism then $\gamma(\pi_1) \equiv \gamma(\pi_2) \pmod{N_2}$. Therefore, $\gamma(\pi'_1) \equiv \gamma(\pi'_2) \pmod{N_2}$ and (M2) holds for θ . ■

17.3.2. Stallings' fold. In this section we define Stallings' folds for pseudo conjugacy graphs. Folding procedure of pseudo conjugacy graphs is an extension of Stallings' folding procedure for X -digraphs in the following sense, if Γ' is a result of a fold of a pseudo conjugacy graph Γ then Γ' (as an X -digraph) is a result of a fold of a pseudo conjugacy graph Γ (as an X -digraph). See [145] for more information on Stallings' folding procedure for X -digraphs.

We say that a pseudo conjugacy graph Γ is *not reduced (not folded)* if there exist distinct edges $e_1 = v \xrightarrow{(x, \gamma_1)} u_1$, $e_2 = v \xrightarrow{(x, \gamma_2)} u_2$ in Γ with the same origin v and the same label x . Otherwise we say that Γ is *reduced (folded)*. The next algorithm performs a Stallings' fold of e_1 and e_2 in Γ .

REMARK 17.3.4. We assume that ∞ is divisible by any positive number $k \in \mathbb{N}$. Therefore, $\gcd(\infty, k) = k$ for any $k \in \mathbb{N}$.

ALGORITHM 17.3.5 (Stallings' fold).

SIGNATURE: $(\Gamma', N', \varphi) = Fold(\Gamma, N, e_1, e_2)$.

INPUT: A weighted X -digraph Γ , $N \in \mathbb{N} \cup \{\infty\}$, and a pair of distinct edges $e_1 = v \xrightarrow{(x, \gamma_1)} u_1$, $e_2 = v \xrightarrow{(x, \gamma_2)} u_2$ in Γ .

OUTPUT: A weighted X -digraph Γ' , $N' \in \mathbb{N} \cup \{\infty\}$ and a canonical epimorphism $\varphi : \Gamma \rightarrow \Gamma'$.

COMPUTATIONS:

A) If $u_1 \neq u_2$ (Figure 17.3):

1) Put

$$u = \begin{cases} u_1, & v \neq u_1; \\ u_2, & \text{otherwise;} \end{cases}$$

and

$$\varepsilon = \begin{cases} \gamma(e_2) - \gamma(e_1), & v \neq u_1; \\ \gamma(e_1) - \gamma(e_2), & \text{otherwise.} \end{cases}$$

- 2) Let $\Gamma_0 = Shift_\varepsilon(\Gamma, u)$ and φ_{Shift} a corresponding X -digraph isomorphism $\Gamma \rightarrow \Gamma_0$.
- 3) Identify the vertices u_1 and u_2 in Γ_0 and denote the result by Γ' . Let $\varphi_1 : \Gamma_0 \rightarrow \Gamma'$ be the corresponding epimorphism.
- 4) Return a triple $(\Gamma', N, \varphi_{Shift} \circ \varphi_1)$.

B) If $u_1 = u_2$:

- 1) If $\gamma(e_1) \neq \gamma(e_2)$ then put $N' = \gcd(N, |\gamma(e_1) - \gamma(e_2)|)$. Otherwise put $N' = N$.
- 2) Identify the edges e_1 and e_2 in Γ into an edge $e = v \xrightarrow{(x, \gamma)} u_1$ and denote the result by Γ' . Let $\varphi : \Gamma \rightarrow \Gamma'$ be the corresponding epimorphism.
- 3) Return a triple (Γ', N', φ) .

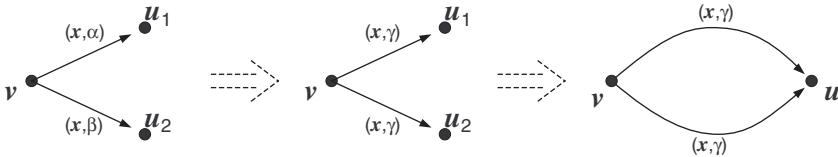


FIGURE 17.3. Fold (Case A).

Note that Case B ($u_1 = u_2$) can be viewed as a removal of the edge e_2 and so, in that case, Γ' is a subgraph of Γ .

PROPOSITION 17.3.6. *Let $(\Gamma', N', \varphi) = Fold(\Gamma, N, e_1, e_2)$. Then*

- (F1)** *(Γ, N) is a pseudo conjugacy graph of w in G if and only if (Γ', N') is.*
(F2) *If (Γ, N) is a pseudo conjugacy graph of w in G then φ is a PCG-morphism.*

Proof. We first show (F1).

“ \Leftarrow ” Clearly, (PCG2) holds for (Γ, N) . We will show that (PCG1') holds for (Γ, N) too. Let π be a loop in Γ and $\pi' = \varphi(\pi)$. Consider two cases.

CASE ($u_1 \neq u_2$). On steps A.1) and A.2) we perform a shift of u_1 or u_2 to make $\gamma(e_1) = \gamma(e_2)$ in Γ_0 . By Proposition 17.3.2 (Γ_0, N) is a pseudo conjugacy graph of $w \in G$ if and only if (Γ, N) is. Hence, we may assume that $\Gamma = \Gamma_0$ and $\gamma(e_1) = \gamma(e_2)$. Clearly, $\mu(\pi) = \mu(\pi')$ and $\gamma(\pi) = \gamma(\pi')$ in Γ_0 . Since (Γ', N') is a pseudo conjugacy graph of w in G ,

$$\mu(\pi) = \mu(\pi') \sim_G w^{\gamma(\pi')} = w^{\gamma(\pi)}.$$

CASE ($u_1 = u_2$). Clearly, $\mu(\pi) = \mu(\pi')$ and

$$\gamma(\pi) = \gamma(\pi') + k|\gamma(e_1) - \gamma(e_2)|$$

where $k \in \mathbb{Z}$ (the last holds since the mapping φ sometimes switches $\gamma(e_2)$ to $\gamma(e_1)$ in Γ'). Since N' divides $|\gamma(e_1) - \gamma(e_2)|$ and (Γ', N') is a pseudo conjugacy graph of w we have

$$\mu(\pi) = \mu(\pi') \sim_G w^{\gamma(\pi')} =_G w^{\gamma(\pi')} w^{k|\gamma(e_1) - \gamma(e_2)|} = w^{\gamma(\pi)}.$$

“ \implies ” Let $\pi' = e'_1, \dots, e'_k$ be a loop in Γ' . To show that $\mu(\pi') \sim_G w^{\gamma(\pi')}$ we will find a path $\pi \in \Gamma$ such that $\mu(\pi) = \mu(\pi')$ in $F(X)$ and $\gamma(\pi) = \gamma(\pi')$.

CASE ($u_1 \neq u_2$). Notice that φ is bijection of edges. So we can define a sequence of edges $\pi_0 = e_1, \dots, e_k \in \Gamma$ where $\varphi(e_i) = e'_i$. Since φ is not bijective on the vertices of Γ (recall $\varphi(u_1) = \varphi(u_2)$) π_0 might be not connected at u_1 and u_2 . To make π_0 connected we fill the gaps in π_0 with $(e_1^{-1}e_2)^{\pm 1}$ depending on a direction of π_0 . Since $\mu(e_1^{-1}e_2) =_{F(X)} \mu(e_2^{-1}e_1) =_{F(X)} 1$ and $\gamma(e_1^{-1}e_2) = \gamma(e_2^{-1}e_1) = 0$ the obtained path π is the required.

CASE ($u_1 = u_2$). We first show that obtained N' satisfies the property (PCG2). Indeed, N' changes to $\gcd(N, |\gamma(e_1) - \gamma(e_2)|)$ only when $\gamma(e_1) \neq \gamma(e_2)$. Consider a loop $l = e_1e_2^{-1}$ in Γ and notice that $\gamma(l) = \gamma(e_1) - \gamma(e_2)$ and $\mu(l) =_F 1$. Finally, since Γ is, by assumption, a pseudo conjugacy graph $1 \sim_G w^{\gamma(l)} = w^{\gamma(e_1) - \gamma(e_2)}$ and $1 = w^N$ which gives us

$$1 = w^{\gcd(N, |\gamma(e_1) - \gamma(e_2)|)}.$$

To finish the proof for this case it remains to show that there exists the claimed above loop c . Since Γ' is a subgraph of Γ we can take π to be a preimage of π' in Γ' . Obviously, it possesses all the claimed properties.

Now we show (F2). Since N' , when changed, becomes $\gcd(N, |\gamma(e_1) - \gamma(e_2)|)$ the property (M1) holds. To prove (M2) consider a loop $\pi \in \Gamma_1$ and its image $\pi' = \varphi(\pi)$. As proved above $\mu(\pi) = \mu(\pi')$ and $\gamma(\pi) = \gamma(\varphi(\pi))$ in the first case and $\gamma(\pi) \equiv \gamma(\pi') \pmod{N'}$ in the second case. Therefore, (M2) is proved and φ is a PCG-morphism. ■

17.3.3. Stallings’ procedure. Now we are ready to present the algorithm which completely folds weighted X -digraphs.

ALGORITHM 17.3.7 (Stallings’ procedure).

SIGNATURE: $(\Gamma', N', \varphi_S) = S(\Gamma, N)$.

INPUT: A weighted X -digraph Γ , $N \in \mathbb{N} \cup \{\infty\}$.

OUTPUT: A weighted X -digraph Γ' , $N' \in \mathbb{N} \cup \{\infty\}$ and a canonical epimorphism $\varphi : \Gamma \rightarrow \Gamma'$.

INITIALIZATION: Put $\Gamma' = \Gamma$, $N' = N$, and $\varphi_S = \text{id}$.

COMPUTATIONS:

- A) While there is a pair of edges $e_1 = v \xrightarrow{(x, \gamma_1)} u_1$, $e_2 = v \xrightarrow{(x, \gamma_1)} u_2$ in Γ' :
 - 1) Compute $(\Gamma', N', \varphi_0) = \text{Fold}(\Gamma', N', e_1, e_2)$.
 - 2) Put $\varphi_S = \varphi_S \circ \varphi_0$.
- B) Return a triple (Γ', N', φ_S) .

Clearly Stallings’ procedure for finite weighted X -digraphs terminates in a finitely many steps since each fold decreases $|V(\Gamma')| + |E(\Gamma')|$ by 1. In contrast to X -digraphs the result of Stallings’ procedure applied to a weighted X -digraph depends on a sequence of Stallings’ folds. One might obtain different values of a weight function γ using different sequences of folds; though the X -digraph structure of a folded Γ does not depend on a particular sequence of folds.

PROPOSITION 17.3.8. *Let G be a group, X a generating set for G , Γ a weighted X -digraph, $N \in \mathbb{N} \cup \{\infty\}$, and $w = w(X)$. If $(\Gamma', N', \varphi_S) = S(\Gamma, N)$ then*

- (S1) (Γ, N) is a pseudo conjugacy graph of w in G if and only if (Γ', N') is.
- (S2) If (Γ, N) is a pseudo conjugacy graph of w in G then φ_S is a PCG-morphism.

Proof. Follows from Proposition 17.3.6. ■

Our next goal is to show that the number N' does not depend on a particular sequence of folds in Algorithm 17.3.8. Define a set of trivial loops in Γ by

$$TL(\Gamma) = \{\pi \mid \pi \text{ is a loop in } \Gamma \text{ s.t. } \mu(\pi) =_{F(X)} 1\}.$$

By a *potential order* of a pseudo conjugacy graph (Γ, N) of w in G we call the following number:

$$(57) \quad PO(\Gamma, N) = \gcd(\{|\gamma(\pi)| \mid \pi \in TL(\Gamma), \gamma(\pi) \neq 0\} \cup \{N\}).$$

LEMMA 17.3.9. *Let (Γ, N) be a folded pseudo conjugacy graph of w in G . Then $N = PO(\Gamma, N)$.*

Proof. Observe that $\{|\gamma(c)| \mid c \in TL(\Gamma), \gamma(c) \neq 0\} = \emptyset$ for a folded Γ . ■

LEMMA 17.3.10. *Let (Γ, N) be a non-folded pseudo conjugacy graph of $w \in G$ and $(\Gamma', N', \varphi) = Fold(\Gamma, N, e_1, e_2)$. Then $PO(\Gamma, N) = PO(\Gamma', N')$.*

Proof. Let $e_1 = v \xrightarrow{(x, \gamma_1)} u_1$ and $e_2 = v \xrightarrow{(x, \gamma_2)} u_2$. Consider two cases ($u_1 \neq u_2$ and $u_1 = u_2$).

If $u_1 \neq u_2$ then $N' = N$ and Γ' is obtained by identification of the vertices u_1 and u_2 . For any loop $\pi \in \Gamma$ (we refer to the proof of Proposition 17.3.6) we have $\gamma(\pi) = \gamma(\varphi(\pi))$ and $\mu(\pi) = \mu(\varphi(\pi))$. On the other hand, for any loop $\pi' \in \Gamma'$ we can choose (see the proof of Proposition 17.3.6) a loop $\pi \in \Gamma$ such that $\gamma(\pi) = \gamma(\pi')$ and $\mu(\pi) = \mu(\pi')$ in $F(X)$. Therefore, $PO(\Gamma, N) = PO(\Gamma', N')$ holds.

If $u_1 = u_2$ then $N' = \gcd(N, |\gamma(e_1) - \gamma(e_2)|)$ and Γ' is obtained from Γ by removing e_2 . For $\pi \in \Gamma$ we have $\gamma(\pi) \equiv \gamma(\varphi(\pi)) \pmod{N'}$ and $\mu(\pi) = \mu(\varphi(\pi))$. Therefore $PO(\Gamma', N')$ divides $PO(\Gamma, N)$. On the other hand, for $\pi' \in \Gamma'$ there exists π such that $\pi' = \varphi(\pi)$ such that $\gamma(\pi) = \gamma(\pi')$ and $\mu(\pi) = \mu(\pi')$ (since Γ' is a proper subgraph of Γ). Finally, notice that $PO(\Gamma, N)$ divides N' since $\gamma(e_1 e_2^{-1}) = \gamma(e_1) - \gamma(e_1)$ and $\mu(e_1 e_2^{-1}) =_{F(X)} 1$. Therefore, the equality $PO(\Gamma, N) = PO(\Gamma', N')$ holds. ■

COROLLARY 17.3.11. *Let (Γ, N) be a pseudo conjugacy graph of w in G and $(\Gamma', N', \varphi_S) = S(\Gamma, N)$. Then $N' = PO(\Gamma, N)$.*

COROLLARY 17.3.12. *Let (Γ, N) be a pseudo conjugacy graph of $w \in G$ and, $(\Gamma_1, N_1, \varphi_1) = S(\Gamma, N)$ and $(\Gamma_2, N_2, \varphi_2) = S(\Gamma, N)$ be two results of Stallings' procedure obtained by different sequences of folds. Then $N_1 = N_2$.*

In other words, the number N of a folded pseudo conjugacy graph does not depend on a sequence of folds. Besides, we know that the X -digraph structure of Γ does not depend on the sequence of folds. The only thing that changes is a weight function γ .

LEMMA 17.3.13. *Let $\tau : (K_1, N_1) \rightarrow (K_2, N_2)$ be a PCG-morphism. Then there exists $c \in \mathbb{N}$ such that $PO(K_1, N_1) = c \cdot PO(K_2, N_2)$ (i.e., $PO(K_2, N_2)$ divides $PO(K_1, N_1)$).*

Proof. An immediate corollary of definitions of a PCG-morphism and a potential order. ■

PROPOSITION 17.3.14. *Stallings' procedure is a functor from the category of pseudo conjugacy graphs of w in G to the category of folded pseudo conjugacy graphs of w .*

Proof. First, we show that if $\tau : K_1 \rightarrow K_2$ is a PCG-morphism, $(K'_1, N'_1, \varphi_1) = S(K_1, N_1)$, and $(K'_2, N'_2, \varphi_2) = S(K_2, N_2)$ then there exists a PCG-morphism $\theta : (K'_1, N'_1) \rightarrow (K'_2, N'_2)$ such that the diagram below commutes.

$$(58) \quad \begin{array}{ccc} (K_1, N_1) & \xrightarrow{\tau} & (K_2, N_2) \\ \downarrow \varphi_1 & & \downarrow \varphi_2 \\ (K'_1, N'_1) & \xrightarrow{\theta} & (K'_2, N'_2) \end{array}$$

We already noticed that Stallings' procedure acts in a usual way on the X -digraph level. Therefore, if the diagram (58) is viewed as the diagram with the corresponding X -digraphs, then it is known that θ exists and unique. We claim that θ is a required PCG-morphism. For that it is sufficient to prove that properties (M1) and (M2) hold.

Property (M1) of θ holds by Lemma 17.3.13. To show (M2) consider a reduced loop π'_1 in K'_1 . The loop π'_1 can be lifted up to a loop $\pi_1 \in K_1$ such that $\varphi_1(\pi_1)$ is π'_1 with, perhaps, some backtracks (we refer to the proof of Proposition 17.3.6). Let $\pi_2 = \tau(\pi_1)$ and $\pi'_2 = \varphi_2(\pi_2)$. The loop π'_2 is $\theta(\pi'_1)$ with backtracks. It remains to show that $\gamma(c'_1) \equiv \gamma(\pi'_2) \pmod{N'_2}$.

Indeed, by Proposition 17.3.8 φ_1 and φ_2 are PCG-morphisms. Since τ is also a PCG-morphism we have

$$\begin{aligned} \gamma(\pi_1) &\equiv \gamma(\pi'_1) \pmod{N'_1}, \\ \gamma(\pi_2) &\equiv \gamma(\pi'_2) \pmod{N'_2}, \\ \gamma(\pi_1) &\equiv \gamma(\pi_2) \pmod{N_2}, \\ \mu(\pi_1) &=_{\mathcal{F}} \mu(\pi'_1), \\ \mu(\pi_2) &=_{\mathcal{F}} \mu(\pi'_2), \\ \mu(\pi_1) &=_{\mathcal{F}} \mu(\pi_2) \end{aligned}$$

Since N'_2 divides N_2 , N_1 and N'_1 ,

$$\gamma(\pi'_1) \equiv \gamma(\pi'_2) \pmod{N'_2}$$

and, hence, (58) commutes. The statement of the proposition easily follows from this. ■

17.3.4. Basic extension.

DEFINITION 17.3.15. Let Γ be a weighted X -digraph, v a vertex in Γ , and $r = r(X)$. Attach a loop of weight 0 labeled with r at the vertex v and denote the result by $\mathcal{E}_r(\Gamma, v)$. The graph $\mathcal{E}_r(\Gamma, v)$ is called a *basic extension* of Γ with r at v . We denote the corresponding canonical X -digraph embedding $\Gamma \hookrightarrow \mathcal{E}_r(\Gamma, v)$ by $\varphi_{\mathcal{E}}$.

PROPOSITION 17.3.16. *Let $G = \langle X; R \rangle$ be a finite presentation, Γ a weighted X -digraph, v a vertex in Γ , $N \in \mathbb{N} \cup \{\infty\}$, $r \in R$, and $w = w(X)$. If $(\Gamma', N) = (\mathcal{E}_r(\Gamma, v), N)$ and $\varphi_{\mathcal{E}} : \Gamma \hookrightarrow \Gamma'$ the canonical embedding then:*

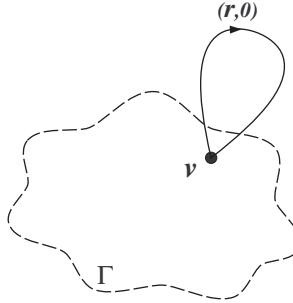


FIGURE 17.4. Graph $\mathcal{E}_r(\Gamma, v)$. Edges of a new loop are consequently labeled with generators x_{i_j} (where $r = x_{i_1} \dots x_{i_m}$) and have weight zero.

- (BE1) (Γ, N) is a pseudo conjugacy graph of w in G if and only if (Γ', N) is.
- (BE2) If (Γ, N) is a pseudo conjugacy graph of w in G then $\varphi_{\mathcal{E}}$ is a PCG-morphism.

Proof. Since N does not change, the property (PCG2) holds for (Γ, N) if and only if it holds for (Γ', N) .

Since Γ is a proper subgraph of Γ' the sufficiency in (BE1) is obvious. Prove the necessity. Denote the added loop in Γ' by l . Let π be a cyclically reduced loop in Γ' . We may assume that π starts (and ends) at v . To show that (PCG1) holds for π we consider two cases. If π does not pass through edges of l then π belongs to Γ which is a conjugacy graph and thus (PCG1) holds for π . If π passes through the edges of l then, since π has no backtracks, π goes all the way along edges of l which has label $r^{\pm 1}$ (depending on a direction) and a weight 0. Therefore, removing an occurrence of l inside of π does not change the weight and the label (as an element of G) of π . The obtained loop is shorter than the initial. Hence, after a few removals of $l^{\pm 1}$ we get a loop from Γ of the same weight and with a label equal to the label of the initial loop (as an element of G). As proved above, (PCG1) holds for such a loop and, hence (BE1) holds.

Finally, by (BE1), if (Γ, N) is a pseudo conjugacy graph of $w \in G$ then (Γ', N) is. Therefore, $\varphi_{\mathcal{E}}$ is a PCG-morphism and (BE2) is done. ■

PROPOSITION 17.3.17. Let $G = \langle X; R \rangle$ be a finite presentation, Γ_1 and Γ_2 be a pseudo conjugacy graphs of $w \in G$ and $\tau : \Gamma_1 \rightarrow \Gamma_2$ be a PCG-morphism. Let v_1 be a vertex in Γ_1 and $v_2 = \tau(v)$ be its image in Γ_2 . Then there exists a PCG-morphism θ such that the following diagram commutes.

$$\begin{array}{ccc} \Gamma_1 & \xrightarrow{\tau} & \Gamma_2 \\ \downarrow \varphi_{\mathcal{E}} & & \downarrow \varphi_{\mathcal{E}} \\ \mathcal{E}_r(\Gamma_1, v_1) & \xrightarrow{\theta} & \mathcal{E}_r(\Gamma_2, v_2) \end{array}$$

Proof. The graph $\mathcal{E}_r(\Gamma_1, v_1)$ is a wedge graph of Γ_1 and a loop l_1 labeled with r . Similarly, the graph $\mathcal{E}_r(\Gamma_2, v_2)$ is a wedge graph of Γ_2 and a loop l_2 labeled with r . Define $\theta : \mathcal{E}_r(\Gamma_1, v_1) \rightarrow \mathcal{E}_r(\Gamma_2, v_2)$ on a subgraph of $\mathcal{E}_r(\Gamma_1, v_1)$ corresponding to Γ_1

to be equal to τ and on l_1 to be l_2 . It is straightforward to check that so defined θ is a PCG -morphism. ■

17.3.5. R -extension algorithm. In this section we present an R -extension algorithm.

ALGORITHM 17.3.18 (R -extension of weighted graphs).

SIGNATURE. $(\Gamma^*, N^*, \varphi_C) = \mathcal{C}(\Gamma, N)$.

INPUT. A finite symmetrized presentation $\langle X; R \rangle$, a weighted X -digraph Γ , and $N \in \mathbb{N} \cup \{\infty\}$.

OUTPUT. A weighted X -digraph Γ^* , an X -digraph morphism $\varphi_C : \Gamma \rightarrow \Gamma^*$, and $N^* \in \mathbb{N} \cup \{\infty\}$.

COMPUTATIONS.

- C1) For each vertex $v \in \Gamma$ and each $r \in R$ add a loop labeled by r of weight 0 at v . Denote the resulting graph by $\mathcal{C}_1(\Gamma)$ and the canonical embedding by $\varphi_{\mathcal{C}_1} : \Gamma \rightarrow \mathcal{C}_1(\Gamma)$.
- C2) Let $(\Gamma^*, N^*, \varphi_S) = S(\mathcal{C}_1(\Gamma), N)$.
- C3) Output a triple $(\Gamma^*, N^*, \varphi_C)$, where $\varphi_C = \varphi_{\mathcal{C}_1} \circ \varphi_S$.

We will denote the graph Γ^* by $\mathcal{C}(\Gamma)$. In Lemmas 17.3.19, 17.3.20, and 17.3.21 we briefly list properties of the R -extension operator \mathcal{C} and the auxiliary operator \mathcal{C}_1 . For proofs we refer the reader to Chapter 16 (see Lemmas 16.3.3, 16.3.4, and 16.3.5).

LEMMA 17.3.19. *Let Γ be a weighted X -digraph and $N \in \mathbb{N} \cup \{\infty\}$. Then the following holds:*

- 1) $\mathcal{C}_1(\Gamma)$ is well-defined, i.e., it does not depend on a sequence of actual transformations in C1).
- 2) $\mathcal{C}(\Gamma)$ (as an X -digraph) and N^* are well-defined, i.e., do not depend on a sequence of actual transformations in C1) and C2).
- 3) If $L(R) > 0$ then $\mathcal{C}_1(\Gamma)$ and $\mathcal{C}(\Gamma)$ are weighted core X -digraphs.
- 4) If $L(R) > 0$ then φ_C is a functor from the category of X -digraphs into the category of folded weighted core X -digraphs.

LEMMA 17.3.20. *Let $\langle X; R \rangle$ be a symmetrized finite presentation and Γ a finite X -digraph. Then the following inequalities hold for the R -extension $\mathcal{C}(\Gamma)$ of Γ :*

- (1) $|V(\mathcal{C}(\Gamma))| \leq (L(R) - |R| + 1)|V(\Gamma)|$,
- (2) $|E(\mathcal{C}(\Gamma))| \leq 2|X| |V(\mathcal{C}(\Gamma))|$.

Moreover, the time complexity of Algorithm 17.3.18 is bounded from above by

$$O(L(R)|V(\Gamma)|\log(L(R)|V(\Gamma)|)).$$

From now on we assume that every letter in X is non-trivially involved in R . Starting with an X -digraph Γ one can iterate the construction above. Put:

$$\mathcal{C}^{(1)}(\Gamma) = \mathcal{C}(\Gamma), \quad \mathcal{C}^{(m+1)}(\Gamma) = \mathcal{C}(\mathcal{C}^{(m)}(\Gamma))$$

Similarly, one can define $\mathcal{C}_1^{(m)}(\Gamma)$ as the result of m consecutive applications of the unary operation \mathcal{C}_1 starting at Γ . As a special case define

$$\mathcal{C}^{(0)}(\Gamma) = S(\Gamma), \quad \mathcal{C}_1^{(0)}(\Gamma) = \Gamma.$$

LEMMA 17.3.21. *Let Γ be an X -digraph. Then the following holds for any non-negative integers m, n :*

- 1) $\mathcal{C}^{(m)}(\Gamma) \simeq S(\mathcal{C}_1^{(m)}(\Gamma));$
- 2) $\mathcal{C}^{m+n}(\Gamma) \simeq \mathcal{C}^n(\mathcal{C}^m(\Gamma));$
- 3) any morphism of weighted X -digraphs $\varphi : \Delta \rightarrow \mathcal{C}^{(m)}(\Gamma)$ gives rise to a morphism $\mathcal{C}^{(n)}(\Delta) \rightarrow \mathcal{C}^{(m+n)}(\Gamma).$

PROPOSITION 17.3.22. Let $G = \langle X; R \rangle$ be a finite symmetrized presentation, Γ a weighted X -digraph, $N \in \mathbb{N} \cup \{\infty\}$, and $w = w(X)$. If $(\Gamma', N', \varphi_C) = S(\Gamma, N)$ then:

- (C1) (Γ, N) is a pseudo conjugacy graph of w in G if and only if (Γ', N') is.
- (C2) If (Γ, N) is a pseudo conjugacy graph of w in G then φ_C is a PCG-morphism.

Proof. Since (Γ', N) is obtained by a sequence of basic extensions of (Γ, N) with relators of G and then by a Stallings' procedure the statement of the proposition follows from Propositions 17.3.16 and 17.3.6. ■

Notice that on the level of X -digraphs Algorithm 17.3.18 works exactly as its counterpart for the word problem (Algorithm 16.3.1).

PROPOSITION 17.3.23. The R -extension operator is a functor from the category of pseudo conjugacy graphs to a category of folded pseudo conjugacy graphs.

Proof. Since R -extension algorithm is a combination of basic extensions and Stallings' fold, the assertion follows from Propositions 17.3.17 and 17.3.14. ■

Let $\langle X; R \rangle$ be a finite presentation and D be a van Kampen diagram over $\langle X; R \rangle$. The diagram D can be viewed as a weighted X -digraph by assigning weight zero to each edge in D . It is straightforward to check that for any $w = w(X)$ the pair (D, ∞) is a pseudo conjugacy graph of $w \in G$.

Let D_0 be a weighted X -digraph consisting of one vertex v_0 and no edges. Let v be a vertex in ∂D starting from which w is read on ∂D . Let $\tau : D_0 \rightarrow D$ be a PCG-morphism such that $\tau(v_0) = v$. The following lemma holds.

LEMMA 17.3.24. Let $m = \delta_v(D)$ the depth of D with respect to the vertex v . There exists a PCG-morphism $\theta : D_0 \rightarrow \mathcal{C}^{(m)}(D_0)$ such that the following diagram commutes:

$$\begin{array}{ccc} (D_0, \infty) & \xhookrightarrow{\tau} & (D, \infty) \\ & \searrow \varphi_C & \downarrow \theta \\ & & \mathcal{C}^{(m)}((D_0, \infty)) \end{array}$$

Therefore, there exists a loop c in $(\Gamma', N') = \mathcal{C}^{(m)}((D_0, \infty))$ such that $\mu(c) = w$ and $\gamma(c) \equiv 0 \pmod{N'}$.

Proof. Similar to a proof of Proposition 16.3.14. ■

17.4. Conjugacy graphs

Let G be a group, $X \subseteq G$ its generating set closed under inversions, and $w = w(X)$. In Section 17.4.1 we show that there exists a conjugacy graph of w in G relative to X and it is unique up to isomorphisms of pseudo conjugacy graphs. Further, in Section 17.4.2, we show that if $\langle X; R \rangle$ is a finite presentation and $G = \langle X; R \rangle$ then the conjugacy graph of w in G can be approximated as a limit of certain weighted X -digraphs.

17.4.1. Existence of conjugacy graphs. Let G be a group, $X \subseteq G$ its generating set closed under inversions, and $w = w(X)$. We will construct a conjugacy graph of w in G directly. Denote by $\Gamma_H(X)$ the Schreier graph of a subgroup H in G relative to the generating set X .

LEMMA 17.4.1. *Let $u = u(X)$. Then $u \simeq_G w^\gamma$ for some $\gamma \in \mathbb{N}$ if and only $\Gamma_{\langle w \rangle}(X)$ contains a loop labeled with u .*

Proof. Indeed, $u \sim_G w^\gamma$ for some $\gamma \in \mathbb{N}$ if and only there exists $s = s(X)$ such that $sus^{-1} =_G w^\gamma$. Let v be the endpoint of a path in $\Gamma_{\langle w \rangle}(X)$ starting at H and labeled with s . By equality $sus^{-1} =_G w^\gamma$ there exists a loop at v labeled with u . ■

Let $\Gamma = \Gamma_{\langle w \rangle}(X)$ and T a spanning tree in Γ . For each vertex $v \in \Gamma$ denote by π_v the path in T connecting the initial vertex (corresponding to $\langle w \rangle$) and v . Define a function $\gamma : E(\Gamma) \rightarrow \mathbb{N}$ as follows. Let $e = v_1 \xrightarrow{x} v_2$. If $e \in T$ then put $\gamma(e) = 0$. If $e \notin T$ then put $\gamma(e) = \gamma$ where $\gamma \in \mathbb{N}$ is the smallest number such that $\mu(p_{v_1} e p_{v_2}^{-1}) = w^\gamma$.

THEOREM 17.4.2. *Let G be a group, $X \subseteq G$ its generating set closed under inversions, and $w = w(X)$. Let $\Gamma = \Gamma_{\langle w \rangle}(X)$ and $N \in \mathbb{N} \cup \{\infty\}$ the order of w in G . Then (Γ, γ, N) is a conjugacy graph of w in G .*

Proof. The property (CG2) is clearly satisfied by the choice of N . Suppose $u \sim w^\gamma$ for some $u = u(X)$ and $\gamma \in \mathbb{N}$. Then by Lemma 17.4.1 there exists a loop π at some vertex v in Γ labeled with u . Let

$$\pi' = p_v \cdot \pi \cdot p_v^{-1}$$

and $e_1^{\varepsilon_1} \dots e_k^{\varepsilon_k}$ a sequence of edges outside of T π' passes (where $\varepsilon_i = \pm 1$ and $e_i = v_{i,1} \xrightarrow{x_i} v_{i,2}$). Clearly,

$$\begin{aligned} \mu(\pi') &=_{F(X)} \mu \left((p_{v_{1,1}} e_1 p_{v_{1,2}}^{-1})^{\varepsilon_1} \dots (p_{v_{k,1}} e_k p_{v_{k,2}}^{-1})^{\varepsilon_k} \right) \\ &= \mu(p_{v_{1,1}} e_1 p_{v_{1,2}}^{-1})^{\varepsilon_1} \dots \mu(p_{v_{k,1}} e_k p_{v_{k,2}}^{-1})^{\varepsilon_k} \\ &=_{G} w^{\varepsilon_1 \gamma(e_1)} \dots w^{\varepsilon_k \gamma(e_k)} =_G w^{\varepsilon_1 \gamma(e_1) + \dots + \varepsilon_k \gamma(e_k)} \end{aligned}$$

and

$$\begin{aligned} \gamma(\pi') &= \gamma \left((p_{v_{1,1}} e_1 p_{v_{1,2}}^{-1})^{\varepsilon_1} \dots (p_{v_{k,1}} e_k p_{v_{k,2}}^{-1})^{\varepsilon_k} \right) \\ &= \varepsilon_1 \gamma(e_1) + \dots + \varepsilon_k \gamma(e_k). \end{aligned}$$

Thus, $\mu(\pi') =_G w^{\gamma(\pi')}$. Finally, notice that $\mu(\pi) = \mu(p_v) \mu(\pi') \mu(p_v)^{-1}$ and $\gamma(\pi) = \gamma(\pi')$. Thus, $\mu(\pi) \sim_G w^{\gamma(\pi)}$.

The argument above can be converted to show that if π is a loop in Γ then $\mu(\pi) \sim_G w^{\gamma(\pi)}$. Thus, (CG1) holds for (Γ, γ, N) . ■

We would like to finish this section with a remark that a conjugacy graph of $w \in G$ is not unique as a weighted graph. For instance, if (Γ, N) is a conjugacy graph and v is a vertex in Γ then an application of a shift operator $Shift_1(\Gamma, v)$ changes the weight function, while not affecting the property of (Γ, N) to be a conjugacy graph. In the next section we show that Γ is unique up to values of a weight function γ .

17.4.2. Conjugacy graph approximation. In this section we show that one can construct the conjugacy graph of $w \in G$ as a limit of a certain sequence of pseudo conjugacy graphs of $w \in G$. The procedure is analogous to Todd-Coxeter algorithm enumerating cosets of the Schreier's graph $\Gamma_{\langle w \rangle}(X)$ of $\langle w \rangle$ relative to X .

Let $G = \langle X; R \rangle$ be a finite symmetrized presentation such that all generators X are non-trivially involved in R and $w = w(X)$. We define a sequence of pseudo conjugacy graphs $\{(\Gamma_i, N_i)\}_{i \in \mathbb{N}}$ as follows. Put

$$(\Gamma_0, N_0) = (\text{Loop}_1(w), \infty)$$

and, recursively, put

$$(\Gamma_{i+1}, N_{i+1}, \varphi_i) = \mathcal{C}(\Gamma_i, N_i)$$

where $\varphi_i : \Gamma_i \rightarrow \Gamma_{i+1}$ is the corresponding canonical morphisms. With φ_i the sequence (Γ_i, N_i) can be viewed as an ascending chain of pseudo conjugacy graphs. Denote the corresponding limit by

$$(\Gamma_\infty, N_\infty) = \lim_{i \rightarrow \infty} (\Gamma_i, N_i).$$

THEOREM 17.4.3 (Approximation of conjugacy graphs). *Let $\langle X; R \rangle$ be a finite symmetrized presentation such that all generators X are non-trivially involved in R , $G = \langle X; R \rangle$, and $w = w(X)$. Then $(\Gamma_\infty, N_\infty)$ is a conjugacy graph of $w \in G$.*

Proof. Let $u = u(X)$ and $\gamma \in \mathbb{N}$ be such that $u =_G s^{-1}w^\gamma s$ for some word $s = s(X)$. Then there exists a van Kampen diagram D over $\langle X; R \rangle$ with a boundary label

$$W = w^\gamma su^{-1}s^{-1}.$$

Let v be a vertex in ∂D starting from which W is read along ∂D and $m = \delta_v(D)$ the depth of D with respect to v . Let D_0 be an X -digraph containing exactly one vertex v_0 and no edges. Let $\tau : D_0 \rightarrow D$ such that $\tau(v_0) = v$ and $\tau_2 : D_0 \rightarrow \text{Loop}_1(w)$ where $\tau(v_0)$ is the initial vertex of $\text{Loop}_1(w)$. By Lemma 17.3.24 there exists a morphism θ and by Proposition 17.3.23 there exists a morphism θ_2 such that the following diagram commutes:

$$\begin{array}{ccc} (D_0, \infty) & \xrightarrow{\tau} & (D, \infty) \\ \downarrow \tau_2 & \searrow \varphi_C & \downarrow \theta \\ (\text{Loop}_1(w), \infty) & & \mathcal{C}^{(m)}((D_0, \infty)) \\ & \searrow \varphi_C & \downarrow \theta_2 \\ & & \mathcal{C}^{(m)}((\text{Loop}_1(w), \infty)) \end{array}$$

where $\mathcal{C}^{(m)}((\text{Loop}_1(w), \infty)) = (\Gamma_m, N_m)$. Therefore, there is a loop π in Γ_m starting at the base vertex v_m of Γ_m such that $\mu(\pi) = w^\gamma su^{-1}s^{-1}$ and $\gamma(\pi) \equiv 0 \pmod{N_m}$. Since Γ_m is folded and contains a loop π_{Γ_m} at v_m such that $\mu(\pi_{\Gamma_m}) = w$ and $\gamma(\pi_{\Gamma_m}) \equiv 1 \pmod{N_m}$ the path π can be decomposed into $c^m p^{-1} t p$, where p is a path starting at v_m and is labeled with s and t is a loop starting at $\beta(s)$ labeled with u^{-1} . Hence,

$$\gamma(\pi) = \gamma(\pi_{\Gamma_m}^m) + \gamma(s) + \gamma(t) - \gamma(s) = m\gamma(\pi_{\Gamma_m}) + \gamma(t) \equiv 0 \pmod{N_m}.$$

Thus, $\gamma(t^{-1}) \equiv m \pmod{N_m}$ and $\mu(t^{-1}) = u$ and the property (CG1) holds. Property (CG2), which requires N to be the order of w , can be shown the same way by considering a diagram with the boundary label w^n . \blacksquare

Observe, that the action of R -extension on pseudo conjugacy graphs coincides with the action of R -extensions on X -digraphs. Therefore, Γ_∞ , if considered as an X -digraph, is the Schreier's graph $\Gamma_{\langle w \rangle}(X)$ of the cyclic subgroup $\langle w \rangle$ in G which is unique up to isomorphism. Moreover, the next proposition holds.

PROPOSITION 17.4.4. *Let (Γ, N) be a conjugacy graph of $w \in G$. Then $N = N_\infty$ and there exists a PCG-morphism $\tau : \Gamma \rightarrow \Gamma_\infty$.*

Proof. By the property (CG1) of conjugacy graphs $N = N_\infty$. Now, we show that the required morphism $\tau : \Gamma \rightarrow \Gamma_\infty$ exists. Let π_Γ be a loop in Γ such that $\mu(\pi_\Gamma) = w$ and $\gamma(\pi_\Gamma) \equiv 1 \pmod{N}$, and v_Γ the initial vertex of π_Γ . If π is a loop in Γ at v_Γ then $\mu(\pi) =_G w^{\gamma(\pi)}$.

Consider Γ_∞ . Let π_{Γ_∞} be the base loop in Γ_∞ (the image of the initial graph $Loop_1(w)$) and v_{Γ_∞} be its initial vertex. It follows from the proof of Theorem 17.4.3 that there is a loop $\pi' \in \Gamma_\infty$ such that $\mu(\pi') = \mu(\pi_\Gamma)$ and $\gamma(\pi') \equiv \gamma(\pi_\Gamma) \pmod{N}$. Therefore, since Γ and Γ_∞ are folded there exists the required PCG-morphism. \blacksquare

PROPOSITION 17.4.5 (On initial approximation). *For any pseudo conjugacy graph (Γ, N) of w in G which contains a loop labeled with w*

$$\lim_{i \rightarrow \infty} \mathcal{C}^{(i)}(\Gamma, N) = (\Gamma_\infty, N_\infty).$$

Proof. By Proposition 17.3.23 the following diagram commutes:

$$\begin{array}{ccc} (Loop_1(w), \infty) & \xrightarrow{\quad} & (\Gamma, N) \\ \downarrow & & \downarrow \\ \mathcal{C}^{(i)}((Loop_1(w), \infty)) & \xrightarrow{\quad} & \mathcal{C}^{(i)}((\Gamma, N)) \end{array}$$

Thus, the result. \blacksquare

This method can be used for computation of the order of a word w in G . If for some m the number N_m is finite then w has a finite order in G which is a divisor of N_m . Of course, the same result can be obtained by variations of coset enumeration (Todd Coxeter algorithm), but our algorithm seems to be more natural and efficient. Though there are no any experimental evidences of this statement. The principal thing is that on each step we have all powers of w in pseudo conjugacy graph. While using the Todd-Coxeter technique each time we have only a limited number of powers of w . So, if w has a large order n , it will take a long time for the Todd-Coxeter algorithm just to construct a path labeled with w^n .

17.5. Annular (Schupp) diagrams

Let $\langle X; R \rangle$ be a finite presentation and $G = \langle X; R \rangle$. An *annular diagram* or *singular annulus* over $\langle X; R \rangle$ is a pair (S, f) , where S is a finite combinatorial annulus, and f a dimension preserving map from S into the Cayley complex $C = C(X; R)$.

Any annular diagram D is bounded by two paths, one of which bounds D from the inner hole and the other bounds D from the outer space. The inner boundary

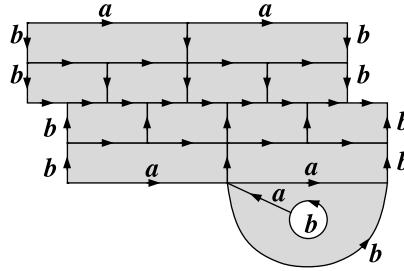


FIGURE 17.5. Example of an annular diagrams over $\langle a, b; a^b = a^2 \rangle$ with boundary labels (read in counterclockwise direction) b and $b^3a^{-1}b^{-2}a^{-2}b^2ab^{-2}a$. Missing labels of horizontal edge are a 's and of vertical edges are b 's.

of D will be denoted by $\partial_{in}D$ and the outer boundary of D will be denoted by $\partial_{out}D$. Thus, $\partial D = \partial_{in}D \cup \partial_{out}D$.

Let D_1 and D_2 be diagrams over $\langle X; R \rangle$. We say that D_1 and D_2 are *isomorphic* if there exists a homeomorphism of the Euclidean plane which induces an isomorphism of corresponding 2-complexes.

PROPOSITION 17.5.1 (Existence of annular diagrams). *Let $G = \langle X; R \rangle$ be a finitely presented group and $w_1 = w_1(X)$, $w_2 = w_2(X)$. Words w_1 and w_2 are conjugate in G if and only if there exists an annular diagram D over $\langle X; R \rangle$ such that $\mu(\partial_{in}D) = w_1$ and $\mu(\partial_{out}D) = w_2$.*

Proof. If w_1 and w_2 are conjugate in G then there exists $v = v(X)$ such that $w_1 = v^{-1}w_2v$ holds in G . Therefore (by van Kampen Lemma) there exists a van Kampen diagram D' over $\langle X; R \rangle$ with the boundary label $w_1^{-1}v^{-1}w_2v$. Sewing D' along v^{-1} and v we obtain the required diagram D .

To show the converse we define p to be a shortest path in D from $\alpha(\partial_{in}D)$ and $\alpha(\partial_{out}D)$ and cut A along p . Denote the result by D' . Since p is a shortest path in D the diagram D' is connected and simply connected. Hence, D' is a van Kampen diagram labeled with $w_1^{-1}v^{-1}w_2v$ and the statement is proved. ■

Let D be an annular diagram over $\langle X; R \rangle$. We will say that $|\partial_{in}D| + |\partial_{out}D|$ is a *perimeter* of D and denote it by $l(D)$.

17.5.1. Depth of annular diagrams.

DEFINITION 17.5.2 (Vertex chain). Let D be an annular diagram over $\langle X; R \rangle$. A sequence of vertices v_1, \dots, v_k is called a *vertex chain* if each pair v_i, v_{i+1} ($i = 1, \dots, k-1$) belongs to some cell or free edge c_i in D . We say that a chain v_1, \dots, v_k has *length* k .

Let K_1 and K_2 be two subcomplexes of D . We say that K_1 and K_2 are connected by a chain in D if there exists a vertex chain $v_1, \dots, v_q \in M$ such that $v_1 \in K_1$ and $v_q \in K_2$. The length of the shortest chain connecting K_1 and K_2 is called the *chain distance* $d(K_1, K_2)$ between K_1 and K_2 .

Recall the definition of depth of 2-complexes. Let K be a subcomplex of a 2-complex L . The number

$$\delta_K(L) = \max\{d(K, \bar{c}) \mid c \in (C(L) \setminus C(K)) \cup (FE(L) \setminus FE(K))\}$$

is called the *depth* of L with respect to K .

DEFINITION 17.5.3 (Depth of an annular diagram). The depth of an annular diagram D is the number

$$\delta(D) = \delta_{\partial D}(D) = \max_{K \in C(D)} \{d(K, \bar{c}) \mid c \in E(\partial D)\}.$$

DEFINITION 17.5.4 (Depth of a pair of words). Let $\langle X; R \rangle$ be a finite presentation, $w_1 = w_1(X)$, and $w_2 = w_2(X)$. Denote by $D(w_1, w_2)$ the set of all annular diagrams over $\langle X; R \rangle$ with boundary labels w_1 and w_2 . Define the depth $\delta(w_1, w_2)$ of a pair (w_1, w_2) as follows:

$$\delta(w_1, w_2) = \begin{cases} \min\{\delta(D) \mid D \in D(w_1, w_2)\} & \text{if } D(w_1, w_2) \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

17.5.2. Annular diagrams as pseudo conjugacy graphs. Let $\langle X; R \rangle$ be a finite presentation and D be an annular diagram over $\langle X; R \rangle$ with boundary labels w_1 and w_2 . In this section we show that D can be viewed as a pseudo conjugacy graph of w_1 or w_2 in G .

Our goal is to construct a weight function for D . Denote by v_0 the initial vertex of $\partial_{in}D$ (to be treated as the base vertex). Let T be a spanning tree for D . For each vertex $v \in V(D)$ denote by p_v a path from v_0 to v in T . Define a function $\gamma_T : E(D) \rightarrow \mathbb{Z}$ the following way. For $e = v_1 \xrightarrow{x} v_2$ put $\gamma(e) = 0$ if $e \in T$ and put $\gamma(e)$ to be the number of times the loop $p_{v_1} e p_{v_2}^{-1}$ goes around the inner hole (in a counterclockwise direction) if $e \notin T$. Denote by D_T the pair (D, γ_T) .

PROPOSITION 17.5.5. *Let D be an annular diagram over $\langle X; R \rangle$ such that $\mu(\partial_{in}D) = w_1$ and $\mu(\partial_{out}D) = w_2$. The pair (D_T, ∞) is a pseudo conjugacy graph of w_1 (and w_2) in G .*

Proof. Clearly, the property (PCG2) holds for (D_T, ∞) . To prove the property (PCG1') consider an arbitrary loop π in D . If $\pi = p_{v_1} e p_{v_2}^{-1}$ for some $e \in E(D) \setminus T$ then, by definition of γ_T , $\mu(\pi) =_G w^{\gamma_T(\pi)}$. If π is some other loop then arguing as in the proof of Theorem 17.4.2 one can show that $\mu(\pi) \sim_G w^{\gamma_T(\pi)}$. ■

17.6. The conjugacy search problem

Pseudo conjugacy graphs can be used to solve the conjugacy search problem in a similar way as approximations of Cayley graphs are used to solve the word search problem. In this section we use ideas of Sections 17.3, 17.4, and 17.5 to construct an algorithm solving the conjugacy search problem.

17.6.1. Annular diagram bisection. In this section we will prove the conjugacy criterion theorem. This theorem will allow us later to express the complexity of the conjugacy search problem in terms of depth of annular diagrams.

First, we show that any annular diagram D can be cut into two annular diagrams D_1, D_2 in a certain way. Denote $\partial_{out}D$ by l_1 and $\partial_{in}D$ by l_2 . Suppose $m = \delta(D)$. Let π be an edge simple, without self-intersections loop in D which goes exactly once around of the annular hole. The loop π cuts D into two annular

diagrams D_1 and D_2 . Since π is edge simple $\partial D_1 = l_1 \cup \pi$ and $\partial D_2 = l_2 \cup \pi$. We say that π cuts D in the middle if $\delta_{l_1}(D_1) \leq m$ and $\delta_{l_2}(D_2) \leq m$, i.e. cells in D_1 and in D_2 are at distance at most m from the outer boundaries l_1 and l_2 , respectively.

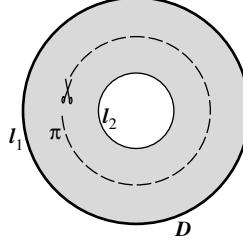


FIGURE 17.6. Middle-cut.

PROPOSITION 17.6.1. *Any annular diagram D over $\langle X; R \rangle$ can be cut in the middle.*

Proof. Let D be an annular diagram over $\langle X; R \rangle$ and $m = \delta(D)$. Define a function α on the set of cells $C(D)$ into the set $\{1, 2\}$ which will specify to which part (D_1 or D_2) cells will belong. If $d(c, \partial D) = 1$ then put $\alpha(c) = 1$ if c touches l_1 (i.e. $\partial c \cap l_1 \neq \emptyset$) and $\alpha(c) = 2$ otherwise. If $d(c, \partial D) = 2$ then put $\alpha(c) = 1$ if $d(c, l_1) = 2$ and $\alpha(c) = 2$ otherwise. Otherwise put $\alpha(c) = 2$, and so on. The function α defines a partition of $C(D)$ into cells with $\alpha = 1$ and $\alpha = 2$. Observe that free edges in D belong to ∂D . Therefore, using α we can define two submaps P_1 and P_2 of D . The submaps P_1 and P_2 are connected, but, in general, not simply connected.

We say that a sequence of cells $\bar{c} = c_1, \dots, c_k$ in D is a *cell-chain* if for each $j = 1, \dots, k - 1$,

$$\partial c_j \cap \partial c_{j+1} \neq \emptyset.$$

The number k is called the length of \bar{c} and is denoted by $|\bar{c}|$. We say that a cell-chain c_1, \dots, c_k is geodesic if k is the smallest length of a cell-chain connecting c_1 and c_k . We will be interested in geodesic chains such that $\partial c_k \cap l_i \neq \emptyset$, i.e., geodesics that connect some cell c_1 with the boundary. Note that any such chain entirely belongs to P_1 or P_2 .

Let $\bar{c} = c_1, c_2, \dots, c_k \in P_i$ ($i = 1, 2$) be a geodesic cell-chain connecting c_k with ∂D . It follows from the definition of α that $d(c_k, l_i) = k$.

Let $\bar{c} = c_1, c_2, \dots, c_k \in P_1$ be a geodesic connecting c_k with l_1 and $\bar{d} = d_1, d_2, \dots, d_m \in P_2$ be a geodesic connecting d_m with l_2 . We claim that they do not intersect, i.e., the situation in Figure 17.7 is impossible.

Assume to the contrary that two geodesic cell-chains \bar{c} and \bar{d} intersect at a vertex v . Let c_s, c_{s+1} and d_t, d_{t+1} be cells in \bar{c} and \bar{d} , respectively, connected to v . Since a part of a geodesic is a geodesic we have $d(c_s, \partial D) = d(c_{s+1}, \partial D) + 1$ and $d(d_t, \partial D) = d(d_{t+1}, \partial D) + 1$. Since all these cells share v we have $d(c_s, \partial D) = d(d_t, \partial D)$ and, hence, $s = t$. But then d_{t+1} must belong to P_1 since it is attached to a cell c_s . Contradiction.

Now, we show that the required loop cutting D in the middle exists. For that we will construct a sequence of loops π_0, \dots, π_m which satisfies the following properties:

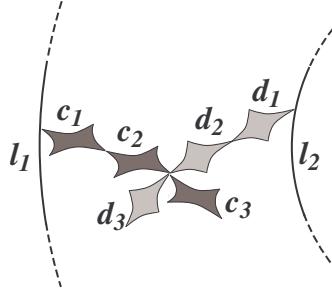
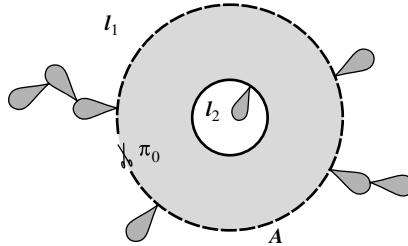


FIGURE 17.7. Intersection of geodesic cell-chains in an annular diagram.

- 1) Each π_i ($i = 0, \dots, m$) is edge-simple and has no self-intersections.
- 2) Each loop π_i cuts D into two annular diagrams A_i (outer) and B_i (inner) such that if φ_i is a projection of A_i into D then $\varphi_i(A_i) \subseteq P_1$.
- 3) Moreover, $\varphi_i(A_i)$ is a proper submap of $\varphi_{i+1}(A_{i+1})$ in P_1 .
- 4) Any geodesic from any cell $c \in \varphi_i(A_i)$ is contained in $\varphi_i(A_i)$.
- 5) Any geodesic from any cell $c \in \varphi_i(B_i) \cap B_i$ is contained in $\varphi_i(B_i)$.

Put $\pi'_0 = l_1$. If $\pi'_0 = e_1 \dots e_k$ is not vertex-simple then it contains a proper subloop $\pi = e_i \dots e_j$ which goes 0 times around the hole. Let c be a cell bounded by π . Clearly, $d(c, l_1) = d(c, \partial A)$ and, hence, $\alpha(c) = 1$. Remove all such subloops from π'_0 and denote the result by π_0 (see Figure 17.8).

FIGURE 17.8. The loop π_0 .

Assume that π_i is already constructed and let P'_1 a set of cells in P_1 that do not belong to $\varphi_i(A_i)$. Let c be a cell in P'_1 with the smallest value $d = d(c, \partial D)$. If $d = 1$ then c touches l_1 at some vertex v . Otherwise, it touches some cell c' which belongs to $\alpha_i(A_i)$ at a vertex v . In both case we perform the following transformations of π_i (see Figures 17.10.b and 17.10.c):

- a) Let $e_1 \dots e_k$ be a boundary ∂c starting from the vertex v .
- b) Insert $e_1 \dots e_k$ in π_i , into the position corresponding to v so that the obtained loop does not intersect itself.
- c) Remove backtracks from the obtained path. Denote the result by π'_{i+1} .

Let $e_i \dots e_j$ be a segment of $e_1 \dots e_k$ and $d_1 \dots d_m$ a segment of π_i remained after removing backtracks. It is possible that a loop π'_{i+1} is not edge-simple (and we

want π'_{i+1} to be edge simple). Consider two cases: ∂c is not edge-simple or ∂c is edge-simple.

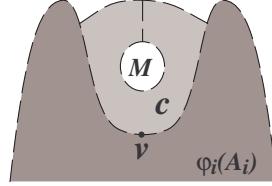


FIGURE 17.9. Case when π'_{i+1} is not edge-simple.

Suppose ∂c is not edge-simple, i.e., touches itself (see Figure 17.9). Since we assume that all relators are cyclically reduced, c bounds a van Kampen diagram, denote it by M . It is easy to see that for each cell c' in M $d(c', \partial D) = d(c', l_1)$ and, therefore, $\alpha(c') = 1$. Let π''_{i+1} be a loop obtained from π'_{i+1} by removing the boundary of M and then removing backtracks. Clearly, π''_{i+1} is edge-simple and the outer diagram which it cuts has projection in P_1 . Denote π''_{i+1} by π_{i+1} .

Suppose that ∂c is edge-simple. Then, since π'_{i+1} is not edge-simple, there are edges d_a and e_b (for some $1 \leq a \leq m$ and $1 \leq b \leq k$) such that $d_a = e_b^{-1}$ (see Figure 17.10).

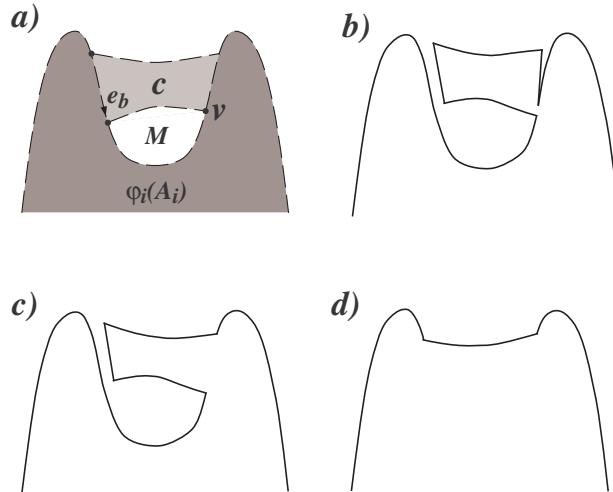


FIGURE 17.10. Case when π'_{i+1} is not edge-simple (figure a). Construction of π_{i+1} (cuts after steps b) and c resp.). Figure d) shows the final result which is obtained by the removal of the internal loop.

Let x be the obtained subloop in π'_{i+1} , M a submap bounded by x , and f a cell in M . Consider two cases. If e_b belongs to l_1 then $d(f, l_1) = 1$. It is easy to see

that $\alpha(f) = 1$ in this event. We remove a subloop x from π'_{i+1} and then remove backtracks from the obtained loop.

Assume that e_b does not belong to l_1 (Figure 17.10). Let c' be a cell such that $c' \neq c$ and $e_b \in \partial c'$. The cell c' belongs to $\varphi_i(A_i)$. Let \bar{c}' be a geodesic cell-chain from c' to l_1 and \bar{c} be a geodesic cell-chain from c to l_1 . Both geodesic chains belong to $\varphi_i(A_i)$ and bound M . Therefore, $\alpha(f) = 1$ (otherwise there would be a geodesic from f to l_2 which intersects some geodesic to l_1). As before, remove a subloop x from π'_{i+1} and then remove backtracks from the obtained loop.

Moreover, if adding c to $\alpha_i(A_i)$ introduces a new “hole” and c touches a cell c' such that $d(c, \partial D) = d(c', \partial D) + 1$ at just a vertex then using the same argument one can show that the hole can be added in $\alpha_i(A_i)$ together with c .

Denote the obtained loop by π_{i+1} . The loop π_{i+1} cuts A into two parts: the outer part A_{i+1} and be the inner part B_{i+1} . Let α_{i+1} be a sewing morphism from A_{i+1} into A . Clearly, the obtained π_{i+1} has no self-intersections and is edge-simple as shown above. Moreover, any geodesic from each cell $c \in \varphi_{i+1}(A_{i+1})$ is contained in $\varphi_{i+1}(A_{i+1})$. ■

PROPOSITION 17.6.2. *Let D be an annular diagram, $l_2 = \partial_{in}D$, $w_2 = \mu(l_2)$, and $m = \delta_{l_1}(A)$. Then there exists a PCG-morphism θ from the pseudo conjugacy graph (D, ∞) of $w_2 \in G$ into $\rightarrow \mathcal{C}^{(m)}((\text{Loop}_1(w_2), \infty))$ such that the diagram below commutes:*

$$\begin{array}{ccc} (\text{Loop}_1(w_2), \infty) & \xrightarrow{\tau} & (D, \infty) \\ \searrow \varphi_c & & \downarrow \theta \\ & & \mathcal{C}^{(m)}((\text{Loop}_1(w_2), \infty)) \end{array}$$

where τ maps $\text{Loop}_1(w_2)$ onto ∂D .

Proof. The proof of this proposition is similar to the proof of Proposition 17.6.1. We say that a diagram D is a “forest on a loop” (see Figure 17.11) if:

- 1) its inner boundary is a vertex-simple loop;
- 2) if $c_1, c_2 \in C(A) \cup FE(A)$ then $|\partial c_1 \cap \partial c_2| \leq 1$.

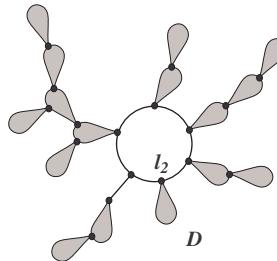


FIGURE 17.11. Forest of cells and free edges on a loop.

First we cut D into an annular diagram which is a “forest on a loop l_2 ”. The loop along which we cut D is constructed in a sequence of steps π_0, \dots, π_k satisfying the following properties:

- 1) Each π_i cuts D into two annular diagrams A_i and B_i .

- 2) A_i is a “forest on a loop l_2 ” diagram.
- 3) If φ_i is a projection of A_i into D then $\varphi_i(A_i)$ is a proper submap of $\varphi_{i+1}(A_{i+1})$.
- 4) If $c \in A_i$ then the chain-distance from c to l_2 in A_i is the same as $d(\varphi_i(c), l_2)$ in D .

Let $\pi_0 = l_2$ – the inner loop of D . The loop π_0 cuts D in two parts: A_0 which is a loop labeled with w_2 ($A_0 = \text{Loop}_1(w_2)$) and $B_0 = D$.

Assume that π_i is the last loop constructed in D . If A_i contains all cells from D then we stop. Otherwise take a cell c which does not belong to A_i with the least value $d = \delta_{l_2}(c)$. If $d = 1$ then c touches l_2 (i.e., $\partial c \cap l_2 \neq \emptyset$) at some vertex v . Let $e_1 \dots e_a = \partial c$ (starting from the vertex v). In this case ($d = 1$) we insert $e_1 \dots e_a$ into the corresponding position of π_i (see Figure 17.10.b). Denote the obtained loop by π_{i+1} .

If $d > 1$ then c touches some cell c' such that $c' \in A_i$ and $\delta_{l_2}(c') = \delta_{l_2}(c) - 1$ at some vertex v . Again, let $e_1 \dots e_a = \partial c$ (starting from the vertex v). Insert $e_1 \dots e_a$ into the corresponding position of π_i . Denote the obtained loop by π_{i+1} .

Clearly, the obtained loop π_{i+1} satisfies all the stated above conditions 1)–4) and the final diagram A_k , denote it by D' , is the required annular diagram. Since D' is obtained by cutting D there exists a sewing morphism $\varphi_1 : D' \rightarrow D$ which maps a boundary l_2 onto itself. Because of the type of D' there exists a graph morphism $\theta' : D' \rightarrow \mathcal{C}^{(m)}(\text{Loop}_1(w_2))$ such that the following diagram commutes:

$$\begin{array}{ccccc} \text{Loop}_1(w_2) & \xrightarrow{\tau} & D' & \xrightarrow{\varphi_1} & D \\ & & \searrow \varphi c & \downarrow \theta' & \\ & & \mathcal{C}^{(m)}(\text{Loop}_1(w_2)) & & \end{array}$$

Since φ_1 is a sewing morphism and $\mathcal{C}^{(m)}(\text{Loop}_1(w_2))$ is folded we can continue θ' through φ_1 . The morphism θ which does that is a required graph morphism. ■

THEOREM 17.6.3 (Conjugacy criterion). *Let D be an annular diagram, $w_1 = \mu(\partial_{out}D)$, $w_2 = \mu(\partial_{in}D)$, and $m = \delta(D)$. Let $\Gamma_1 = \text{Loop}_1(w_1)$ and $\Gamma_2 = \text{Loop}_1(w_2)$. Then there exist equally labeled paths $p_1 \in \mathcal{C}^{(m)}((\Gamma_1, \infty))$ and $p_2 \in \mathcal{C}^{(m)}((\Gamma_2, \infty))$ of weight 1 (i.e., $\mu(p_1) = \mu(p_2)$ and $\gamma(p_1) = \gamma(p_2) = 1$).*

Conversely, if for w_1, w_2 there exists a number m such that $\mathcal{C}^{(m)}((\Gamma_1, \infty))$ and $\mathcal{C}^{(m)}((\Gamma_2, \infty))$ contain paths p_1 and p_2 as above then $w_1 \sim_G w_2$.

Proof. Let D be an annular diagram of depth m . Let π cuts D in the middle into annular diagrams D_1, D_2 such that $\delta_{l_i}(D_i) \leq m$ ($i = 1, 2$). By Proposition 17.6.1 $\partial D_1 = l_1 \cup \pi$ and $\partial D_2 = l_2 \cup \pi$. By Proposition 17.6.2 there exist graph morphisms $D_i \rightarrow \mathcal{C}^{(m)}(\text{Loop}_1(w_i))$ ($i = 1, 2$). Images of c in $\mathcal{C}^{(m)}(\text{Loop}_1(w_i))$ are the required paths p_1 and p_2 . ■

17.6.2. Conjugacy search algorithm. In this section, using the ideas of the previous section, mostly Theorem 17.6.3, we design an algorithm for solving the conjugacy search problem. At the end of the section we give estimates for time complexity of a proposed algorithm.

By Theorem 17.6.3 if pseudo conjugacy graphs $\Gamma_1 = \mathcal{C}^{(m)}(\text{Loop}_1(w_1))$ and $\Gamma_2 = \mathcal{C}^{(m)}(\text{Loop}_1(w_2))$ contain equally labeled loops p_1 and p_2 resp. of weight 1

then $w_1 \sim_G w_1$. The next lemma shows how to find a conjugator for w_1 and w_2 in this situation. Let π_{Γ_i} (where $i = 1, 2$) be a base loop in Γ_i such that $\mu(\pi_{\Gamma_i}) = w_i$ and $v_{\Gamma_i} = \alpha(\pi_{\Gamma_i})$.

LEMMA 17.6.4. *Let q_1 be a path from $\alpha(p_1)$ to v_1 and q_2 a path from $\alpha(p_2)$ to v_2 and $x = \mu(q_2)^{-1}\mu(q_1)$. Then $w_1 =_G x^{-1}w_2x$.*

Proof. Since $\gamma(p_1q_1\pi_{\Gamma_1}^{-1}q_1^{-1}) = 0$ and $\gamma(p_2q_2\pi_{\Gamma_2}^{-1}q_2^{-1}) = 0$ we have

$$\mu(p_1)\mu(q_1)w_1^{-1}\mu(q_1)^{-1} =_G 1 \text{ and } \mu(p_2)\mu(q_2)w_2^{-1}\mu(q_2)^{-1} =_G 1.$$

Finally, since $\mu(p_1) = \mu(p_2)$ we have

$$\mu(q_1)w_1\mu(q_1)^{-1} =_G \mu(q_2)w_2\mu(q_2)^{-1}$$

and, hence, $w_1 =_G \mu(q_1)^{-1}\mu(q_2)w_2\mu(q_2)^{-1}\mu(q_1)$. ■

Next, we show that there exists an effective algorithm for loops p_1 and p_2 (with stated above properties) when they exist in Γ_1 and Γ_2 , respectively. The next definition is a generalization of a product of X -digraphs.

DEFINITION 17.6.5 (Product of weighted X -digraphs). Let Γ_1 and Γ_2 be two weighted X -digraphs. Let $\Gamma = \Gamma_1 \times \Gamma_2$ be a product of Γ_1 and Γ_2 as X -digraphs (see [145]) and let $\pi_1 : \Gamma \rightarrow \Gamma_1$ and $\pi_2 : \Gamma \rightarrow \Gamma_2$ be projection functions. A *product graph* of Γ_1 and Γ_2 is an X -digraph Γ with a weight functions $\gamma_{\Gamma} = (\gamma_1, \gamma_2)$ defined on $e \in \Gamma$ the following way:

$$\gamma_{\Gamma}(e) = (\gamma_{\Gamma_1}(\pi_1(e)), \gamma_{\Gamma_2}(\pi_2(e))).$$

Observe, that $\Gamma_1 \times \Gamma_2$ is not a weighted X -digraph as it is defined in Section 17.2.1 (since the range of γ_{Γ} is $\mathbb{N} \times \mathbb{N}$). Nevertheless, one can extend the function γ_{Γ} on paths in Γ as follows; for $p = e_1 \dots e_k$ (where $e_i = (e'_i, e''_i)$) put $\gamma_{\Gamma}(p) = (\sum_{i=1}^k \gamma_{\Gamma_1}(e'_i), \sum_{i=1}^k \gamma_{\Gamma_2}(e''_i))$. So, we may think of Γ as of a weighted X -digraph.

Let $\Gamma = \Gamma_1 \times \Gamma_2$, $v = (v_1, v_2)$ be a vertex in Γ . Denote the set of all loops in Γ starting at v by L_v . Define the set of indices of L_v by

$$I_v = \{\gamma_{\Gamma}(\pi) \mid \pi \in L_v\}.$$

LEMMA 17.6.6 (Weights of loops in a product of weighted X -digraphs). *The following holds:*

- 1) *The set I_v is an ideal in $\mathbb{Z} \times \mathbb{Z}$.*
- 2) *If vertices u and v are connected in Γ then $I_u = I_v$.*

Proof. Immediately follows from the additivity of the weight function. ■

PROPOSITION 17.6.7. *Weighted graphs Γ_1 and Γ_2 have equally labeled loops of weight 1 if and only if $(1, 1) \in I_v$ for some vertex $v = (v_1, v_2) \in \Gamma_1 \times \Gamma_2$.*

Proof. “ \Rightarrow ” Let c_1 and c_2 be two cycles in Γ_1 and Γ_2 respectively such that $u = \mu(c_1) = \mu(c_2)$ and $\gamma_{\Gamma_1}(c_1) = \gamma_{\Gamma_2}(c_2) = 1$. Let $v_1 = \alpha(c_1)$ and $v_2 = \alpha(c_2)$. Clearly, there exists a loop in $\Gamma = \Gamma_1 \times \Gamma_2$ starting (v_1, v_2) labeled with u of weight $(1, 1)$. Therefore $(1, 1) \in I_v$.

“ \Leftarrow ” Reverse the argument above. ■

Now, our goal is to define an effective algorithm to compute I_v . Let $\Gamma = \Gamma_1 \times \Gamma_2$ and $v = (v_1, v_2)$ a vertex in Γ . Denote by Γ_v a connected component of Γ containing v . Let T be a spanning tree in Γ_v , E' be a subset of edges of Γ_v outside of T . For

each $v' \in \Gamma_v$ denote by $p_{v'}$ the shortest path in T with the origin v and the terminus v' . Define a set X_v to be the set of cycles

$$X_v = \{p_{\alpha(e)}ep_{\beta(e)}^{-1} \mid e \in E'\}.$$

LEMMA 17.6.8 (Generators of I_v). *For any $v \in \Gamma = \Gamma_1 \times \Gamma_2$,*

$$I_v = \langle \gamma_\Gamma(\pi) \mid \pi \in X_v \rangle.$$

Proof. The set X_v generates the fundamental group $\pi_0(\Gamma)$ of Γ (see [145]). Hence, any cycle in Γ_v is a product of loops from X_v and their inverses. So, the result follows from additivity of the weight function γ_Γ . ■

To determine whether a pair $(1, 1)$ belongs to I_v (for some $v = (v_1, v_2) \in \Gamma_1 \times \Gamma_2$) one can apply the Gauss elimination procedure to pairs X_v to compute the generating set for I_v . It is hard to estimate the complexity of this procedure since we do not know a priori the values $\gamma_\Gamma(\pi)$ generating I_v . We will give the final estimate up to the complexity of computation of the generating set for I_v . We would like to point out that any non-trivial value $\gamma_\Gamma(\pi)$ is a valuable information about w_1 and w_2 by itself.

The next algorithm checks whether two pseudo conjugacy graphs of w_1 and w_2 contain equally labeled loops of weight 1 and if they do then finds a conjugator for w_1 and w_2 .

ALGORITHM 17.6.9 (Common cycle of weight 1).

INPUT: Pseudo conjugacy graphs Γ_1 and Γ_2 of w_1 and w_2 correspondingly with fixed base loops c_{Γ_1} and c_{Γ_2} labeled with w_1 and w_2 of weight 1.

OUTPUT: Output *Yes* if a common cycle of weight 1 exists. Otherwise output *No*. COMPUTATIONS:

- A) Compute $\Gamma = \Gamma_1 \times \Gamma_2$.
- B) For each connected component Γ_v (where $v = (v_1, v_2)$):
 - 1) Compute the generating set $\{\gamma_\Gamma(\pi) \mid \pi \in X_v\}$ for I_v .
 - 2) Check if I_v contains $(1, 1)$.
 - 3) If it does then compute paths $p_1 \in \Gamma_1$ from $\alpha(c_{\Gamma_1})$ to v_1 and $p_2 \in \Gamma_2$ from $\alpha(c_{\Gamma_2})$ to v_2 . Output *Yes* and a word $\mu(p_2)\mu(p_1)^{-1}$
- C) Output *No*.

PROPOSITION 17.6.10. *Let Γ_1 and Γ_2 be pseudo conjugacy graphs of w_1 and w_2 , respectively, and $n_1 = |V(\Gamma_1)|$, $n_2 = |V(\Gamma_2)|$. Then Algorithm 17.6.9 terminates in at most Cn_1n_2 steps (for some constant C) up to operation B.2).*

Proof. Consider Algorithm 17.6.9 step by step. To construct $\Gamma = \Gamma_1 \times \Gamma_2$ it is required $C_1n_1n_2$ steps. Now in each component Γ_v it takes $|\Gamma_v|$ steps to construct a spanning tree and compute a generating set X_v for I_v . ■

Equipped with Algorithm 17.6.9 we can present an algorithm for the conjugacy search problem.

ALGORITHM 17.6.11 (Conjugacy search algorithm \mathcal{A}_C).

INPUT: A finite symmetrized presentation $G = \langle X; R \rangle$, and words $w_1 = w_1(X)$ and $w_2 = w_2(X)$.

OUTPUT: If $w_1 \sim_G w_2$ then output *Yes* with a conjugator x (where $w_1 = x^{-1}w_2x$). Otherwise do not stop.

COMPUTATIONS:

- A) Put $n = 0$, $(\Gamma_0, N_0) = (\text{Loop}_1(w_1), \infty)$, and $(\Delta_0, N'_0) = (\text{Loop}_1(w_2), \infty)$.
- B) Apply Algorithm 17.6.9 to graphs Γ_n and Δ_n .
- C) If the answer is *No* then
 - 1) Compute $(\Gamma_{n+1}, N_{n+1}) = \mathcal{C}(\Gamma_n, N_n)$.
 - 2) Compute $(\Delta_{n+1}, N'_{n+1}) = \mathcal{C}(\Delta_n, N'_n)$.
 - 3) Increment n .
 - 4) Goto B).
- D) If the answer is *(Yes, x)* then output *(Yes, x)*.

THEOREM 17.6.12. *Let $G = \langle X; R \rangle$ be a finite presentation, w_1 and w_2 be words in generators $X^{\pm 1}$. Let $m = \delta(w_1, w_2)$ then Algorithm 17.6.11 requires $C|w_1||w_2|L(R)^{2m}$ (for some constant C) steps to recognize w_1 and w_2 as conjugated (in case $m = \infty$ Algorithm 17.6.11 does not stop) and find a conjugator.*

Proof. Follows from Lemma 17.3.20 and Proposition 17.6.10. ■

17.7. Random annular diagrams

In this section we define a notion of a random annular diagram using techniques developed for random van Kampen diagrams in Chapter 16. Recall how we introduced a discrete probability measure on diagrams using the random generators producing random diagrams, so the probability of a diagram was the probability of its generation.

First, recall some definitions from above. Let $\mathcal{K} = \{D_i \mid i \in \mathbb{N}\}$ be a countable (enumerable) collection of diagrams. In here we assume that diagrams from \mathcal{K} are annular diagrams over some fixed presentation $\langle X; R \rangle$ equipped, perhaps, with some extra predicates. Denote by $B : \mathcal{K} \rightarrow \mathcal{K}$ a stochastic map which with probability $p_{i,j}$ maps D_i into D_j . Sometimes we write $B(D_i) = D_j$ if $p_{i,j} > 0$. The map B can be viewed as a random walk on \mathcal{K} defined by the infinite stochastic matrix $(p_{i,j})$. We say that B is a *basic extension* if for every diagrams D_i, D_j such that $B(D_i) = D_j$ there exists a diagram morphism $D_i \rightarrow D_j$.

Given a basic extension B and a diagram, say $D_0 \in \mathcal{K}$, one can define the *transition tree* $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$ of B as follows.

- 1) $V(\mathcal{T}) = \{D_{i_0}, \dots, D_{i_k} \mid D_{i_j} = B(D_{i_{j-1}}) \text{ for each } 1 \leq j \leq k \text{ and } D_{i_0} = D_0\}.$
- 2) $E(\mathcal{T}) = \{(D_{i_0}, \dots, D_{i_k}, D_{i_0}, \dots, D_{i_{k+1}})\}.$
- 3) Each edge $e = (D_{i_0}, \dots, D_{i_k}, D_{i_0}, \dots, D_{i_{k+1}}) \in E(\mathcal{T})$ has an associated number $p_{i_k, i_{k+1}}$.

The vertex ε (empty sequence of diagrams from \mathcal{K}) in the tree \mathcal{T} is called a *root* of \mathcal{T} . Denote by \mathcal{W} the random walk on \mathcal{T} which starts at ε with probability 1.

For $D \in \mathcal{K}$ we denote by $\Phi = \Phi_B(D)$ the set of all diagrams C in \mathcal{K} such that $C = B^n(D)$ for some $n \in \mathbb{N}$. A basic extension B is called \mathcal{K} -*complete* if there exists $D \in \mathcal{K}$ such that $\Phi_B(D) = \mathcal{K}$. More generally, if $\varphi : \mathcal{K} \rightarrow \mathcal{L}$ is a mapping from \mathcal{K} onto a collection of diagrams \mathcal{L} then we say that B is \mathcal{L} -*complete relative to φ* if $\varphi(\Phi) = \mathcal{L}$.

Let $D \in \mathcal{K}$ and v is a vertex in D . We say that B is *locally stable* v if the neighborhood of v eventually stabilizes, i.e., for any infinite sequence of diagrams

$$D = D_{i_1}, D_{i_2}, \dots$$

such that $D_{i_{j+1}} = B(D_{i_j})$, there exists $k \in \mathbb{N}$ such that $N_{D_{i_j}}(v) = N_{D_{i_k}}(v)$ for every $j \geq k$. A random generator B is called *locally stable* if it is stable at every vertex v of every diagram $D \in \mathcal{K}$.

An infinite directed path in \mathcal{T} starting at ε is called a *trajectory*. Let Λ is the set of all trajectories in \mathcal{T} . A *cone* of $\theta \in V(\mathcal{T})$ is the set of all trajectories passing through θ :

$$Cone(\theta) = \{\lambda \in \Lambda \mid \lambda \text{ is passing through } \theta\}.$$

Let \mathcal{F} be a σ -algebra generated by all cones $Cone(\theta)$, where $\theta \in V(\mathcal{T})$. For each $\theta = D_{i_0}, \dots, D_{i_k} \in V(\mathcal{T})$ the real number $P(Cone(\theta))$ is defined as the probability to hit the vertex $\theta \in \mathcal{T}$ by the random walk W , i.e.,

$$P(Cone(\theta)) = \prod_{j=1}^k p_{i_{j-1}, i_j}.$$

By the Kolmogorov's extension theorem the function P extends onto the σ -algebra \mathcal{F} in such a way that $(\Lambda, \mathcal{F}, P)$ is a probability space, so P is a probability measure on Λ .

A random variable $Q : \Lambda \rightarrow \mathbb{N}$ is called a *termination condition*. For a termination condition Q denote by V_Q the set of all stop-vertices of \mathcal{W} in \mathcal{T} relative to the termination condition Q and P_Q the discrete probability measure on V_Q induced from $(\Lambda, \mathcal{F}, P)$. We say that a sequence of termination conditions $\mathcal{Q} = \{Q_i\}_{i \in \mathbb{N}}$ is *complete* for \mathcal{W} if

$$V(\mathcal{T}) = \bigsqcup V_{Q_i}.$$

Let \mathcal{Q} be a complete sequence of termination conditions for \mathcal{W} . Define an asymptotic density of subsets of $V(\mathcal{T})$. Namely, if $S \subseteq V(\mathcal{T})$ then the asymptotic density $\rho_{V(\mathcal{T})}(S)$ of S in $V(\mathcal{T})$ relative to B , D_0 , and \mathcal{Q} is equal to the following limit (if it exists):

$$\rho_{V(\mathcal{T})}(S) = \lim_{i \rightarrow \infty} P_{Q_i}(S \cap V_{Q_i}).$$

If $\mu : \mathbb{N} \rightarrow \mathbb{R}$ is a probability distribution on \mathbb{N} then one can define a discrete probability measure on $V(\mathcal{T})$,

$$P_{V(\mathcal{T})} : V(\mathcal{T}) \rightarrow \mathbb{R}$$

(relative to B , D_0 , \mathcal{Q} and μ) as follows. For $\theta \in V(\mathcal{T})$ such that $\theta \in V_{Q_i}$ for some $i \in \mathbb{N}$ put

$$(59) \quad P_{V(\mathcal{T})}(\theta) = \mu(i)P_{Q_i}(\theta).$$

Using the discrete probability and asymptotic density defined on $V(\mathcal{T})$ one can induce the discrete probability measure on \mathcal{K} and asymptotic density on subsets of \mathcal{K} . For $\theta = D_{i_0}, \dots, D_{i_k} \in \mathcal{T}$ denote by $D(\theta)$ the diagram D_{i_k} . Now, for $D \in \mathcal{K}$ we define a function

$$P_{\mathcal{K}}(D) = \sum_{\theta \in V(\mathcal{T}), D(\theta)=D} P_{V(\mathcal{T})}(\theta).$$

It is easy to see that $P_{\mathcal{K}}$ is a discrete probability measure on \mathcal{K} . To define asymptotic density on \mathcal{K} , define auxiliary sets

$$\mathcal{K}_i = D(V_{Q_i}) = \{D(\theta) \mid \theta \in V_{Q_i}\}$$

with functions $P_{\mathcal{K}_i} : \mathcal{K}_i \rightarrow \mathbb{R}$ such for $D \in \mathcal{K}_i$:

$$P_{\mathcal{K}_i}(D) = \sum_{\theta \in V_{Q_i} \text{ and } D(\theta)=D} P_{Q_i}(\theta).$$

The function $P_{\mathcal{K}_i}$ is a discrete probability measure on \mathcal{K}_i . Moreover, if the sets \mathcal{K}_i form a partition of \mathcal{K} then one can define an *asymptotic density* of diagrams from \mathcal{K} as follows. If $S \subseteq \mathcal{K}$ then

$$\rho_{\mathcal{K}}(S) = \lim_{i \rightarrow \infty} P_{\mathcal{K}_i}(\mathcal{K}_i \cap S)$$

(if it exists) is an asymptotic density of S in \mathcal{K} .

In a similar way one can induce a discrete probability measure $P_{\mathcal{L}}$ on \mathcal{L} . Define a partition of \mathcal{L} into the sets \mathcal{L}_i , define a discrete probability measure on \mathcal{L}_i , and, finally, if \mathcal{L}_i is a partition of \mathcal{L} , introduce an asymptotic density $\rho_{\mathcal{L}}$ on sets from \mathcal{L} .

Finally, define two particular complete series of termination conditions. The first function Q_n simply counts the number of applications of a basic extension. For $\lambda \in \Lambda$ and $n \in \mathbb{N}$ put

$$(60) \quad Q_n(\lambda) = n.$$

The termination conditions of the second type measure the size of a constructed diagram. To introduce it we need the following auxiliary random variable. For $\lambda \in \Lambda$ and $n \in \mathbb{N}$ put

$$X_n(\lambda) = \min\{i \in \mathbb{N} \mid \chi(D(\lambda_i)) = n\}.$$

The second series of termination conditions is defined by

$$(61) \quad \hat{Q}_n(\lambda) = \max\{i \in \mathbb{N} \mid \chi(D(\lambda_i)) = n\} = X_{n+1}(\lambda) - 1.$$

The last series $\{\hat{Q}_n\}$ of termination conditions defines partitions of \mathcal{K} and \mathcal{L} . We will analyze properties of random diagrams relative to $\{\hat{Q}_n\}$.

17.7.1. Basic random extension of annular diagrams. Let $\langle X; R \rangle$ be a finite presentation, $G = \langle X; R \rangle$, and $w = w(X)$. Denote by $\mathcal{L}_w = \mathcal{L}_w(X, R)$ a set of representatives (up to isomorphism) of all annular diagrams D over $\langle X; R \rangle$ in which there exists a loop without self-intersections labeled with w . Any loop π in D with defined above properties will be called a *base loop* in D . Since $\langle X; R \rangle$ is finite \mathcal{L}_w is a countable set. So, let $\mathcal{L}_w = \{D_0, D_1, \dots\}$ be a numeration of diagrams from \mathcal{L}_w . We will always denote by D_0 an annular diagram $Loop(w)$, which is a loop labeled with w . When the index of a diagram $D_i \in \mathcal{L}_w$ is irrelevant it will be omitted, we will refer to D_i simply as to $D \in \mathcal{L}_w$.

PROPOSITION 17.7.1. *Let $D \in \mathcal{L}_w$ and $w_1 = \mu(\partial_{in} D)$ and $w_2 = \mu(\partial_{out} D)$. Then $w \simeq_G w_1 \simeq_G w_2$.*

Proof. Obvious. ■

PROPOSITION 17.7.2. *Let $D_i \in \mathcal{L}_w$. There exist annular diagrams $D_i^{(1)}$ and $D_i^{(2)}$ over $\langle X; R \rangle$ such that $\mu(\partial_{in} D_i^{(1)}) = \mu(\partial_{in} D_i^{(2)}) = w$ and*

$$(62) \quad D_i = D_i^{(1)} \bigvee_{\partial_{in} D_i^{(1)} = \partial_{in} D_i^{(2)}} D_i^{(2)}.$$

Moreover, for any pair of diagrams $D_i^{(1)}$ and $D_i^{(2)}$ such that

$$\mu(\partial_{in}D_i^{(1)}) = \mu(\partial_{in}D_i^{(2)}) = w$$

(62) belongs to \mathcal{L}_w .

Proof. Let π_{D_i} be a base loop in D_i . Cut D_i along π_{D_i} . Let D_{in} and D_{out} be two obtained annular diagrams (inner and outer, respectively). By turning D_{in} inside out we can make $\mu(\partial_{in}D_{in}) = \mu(\partial_{in}D_{out}) = w$. Clearly, D_{in} and D_{out} are the required diagrams. The second part of the statement is obvious. ■

An *extended diagram* D over $\langle X; R \rangle$ is a quintuple $(D^{(1)}, D^{(2)}, M(D^{(1)}), M(D^{(2)}), A)$ where:

- D_1 and D_2 are annular diagrams over $\langle X; R \rangle$ such that $\mu(\partial_{in}D_1) = \mu(\partial_{in}D_2)$;
- $M(D^{(i)}) \subseteq V(D^{(i)})$ (where $i = 1, 2$) is called a set of *marked vertices* (worked out vertices);
- A set

$$A \subseteq (\partial_{out}D_1 \cup \partial_{out}D_2) \setminus (M(D_1) \cup M(D_2))$$

is such that $|A| \leq 1$. We refer to vertices from A as to “active vertices” (vertices in the working).

For an extended diagram $D = (D^{(1)}, D^{(2)}, M(D^{(1)}), M(D^{(2)}), A)$ over $\langle X; R \rangle$ denote by \overline{D} the diagram

$$\overline{D} = D^{(1)} \bigvee_{\partial_{in}D^{(1)} = \partial_{in}D^{(2)}} D^{(2)}.$$

Clearly, \overline{D} is an annular diagram over $\langle X; R \rangle$. Denote by $\mathcal{K}_w = \mathcal{K}_w(X; R)$ the set of all extended annular diagrams D over $\langle X; R \rangle$ such that $\mu(\partial_{in}D_1) = \mu(\partial_{in}D_2) = w$. Clearly, for any $D \in \mathcal{K}_w$ $\overline{D} \in \mathcal{L}_w$. Conversely, for any $D' \in \mathcal{L}_w$ there exists $D \in \mathcal{K}_w$ such that D' and \overline{D} are isomorphic. Morphisms of extended diagrams D are morphisms of annular diagrams \overline{D} that preserve the marked vertices.

Let $S = (s_1, s_2, s_3, s_4)$ be a sequence of reals such that $s_1 + s_2 + s_3 = 1$ and $s_i \in [0, 1]$. The following procedure provides the basic extension B_S of extended annular diagrams.

ALGORITHM 17.7.3 (Basic Extension B_S).

INPUT: Let D be an extended annular diagram over $\langle X; R \rangle$ such that either $A(D) \neq \emptyset$ or $(\partial_{out}D^{(1)} \setminus M(D^{(1)})) \cup (\partial_{out}D^{(2)} \setminus M(D^{(2)})) \neq \emptyset$.

OUTPUT: Diagram $D' = B_S(D)$.

COMPUTATIONS:

- 1) If $|A(D)| = 1$ then take the only vertex $v \in A(D)$. If $|A(D)| = 0$ then, choose randomly and uniformly an unmarked vertex

$$v \in (\partial_{out}D^{(1)} \setminus M(D^{(1)})) \cup (\partial_{out}D^{(2)} \setminus M(D^{(2)})),$$

and put $A(D) = \{v\}$.

- 2) If $|(\partial_{out}D^{(1)} \setminus M(D^{(1)})) \cup (\partial_{out}D^{(2)} \setminus M(D^{(2)}))| > 1$ then with probability s_1 do a , with probability s_2 do b , and with probability $s_3 = 1 - s_1 - s_2$ do c) below:

- a) Take randomly and uniformly a relator $r \in R$. Make a cell N with the boundary label r starting at some vertex $u \in \partial N$. Attach N at v from the outer side by identifying the vertices u and v . Go to 5).
- b) Take randomly and uniformly a letter $y \in X^{\pm 1}$. Make a free edge $e = (u_1, u_2)$ with the label y . Attach e at v from the outer side by identifying the vertices v and u_1 . Go to 5).
- c) Do not attach anything to v and go to 4).
- 3) If $|(\partial_{out}D^{(1)} \setminus M(D^{(1)})) \cup (\partial_{out}D^{(2)} \setminus M(D^{(2)}))| = 1$ and $s_1 + s_2 \neq 0$ then with probability $\frac{s_1}{s_1 + s_2}$ do a) below, otherwise do b):
 - a) Take randomly and uniformly a relator $r \in R$. Make a cell N with the boundary label r starting at some vertex $u \in \partial N$. Attach N at v from the outer side by identifying the vertices u and v . Go to 5).
 - b) Take randomly and uniformly a letter $y \in X^{\pm 1}$. Make a free edge $e = (u_1, u_2)$ with the label y . Attach e at v from the outer side by identifying the vertices v and u_1 . Go to 5).
- 4) a) Let $(e_1, h_1), \dots, (e_k, h_k)$ be all pairs of edges incident to v and such that for each i the following conditions hold:
 - the path $e_i h_i$ belongs to the outer boundary of the diagram (with respect to a fixed orientation);
 - all endpoints of e_i and f_i are unmarked;
 - e_i and h_i^{-1} have the same labels (potential fold);
 - edges e_i and h_i are not free.
 Then for each $i = 1, \dots, k$ with a fixed probability s_4 fold e_i and h_i^{-1} .
 - b) add v to $M(D^{(i)})$,
 - c) remove v from $A(D)$. Go to 5).
- 5) Denote the resulting diagram by $B_S(D)$. Output $B_S(D)$.

The basic extension B_S presented here is similar to the basic extension B_S (Algorithm 16.6.1) from Chapter 16. It is straightforward to check that B_S has the following basic properties (they are all similar to the properties of B_S in Algorithm 16.6.1).

LEMMA 17.7.4 (Properties of B_S). *Let $D^* = B_S(D)$ and $i = 1, 2$. Then:*

- 1) *If a vertex $v \in D^{(i)}$ is marked then $N_{D^{(i)}}(v) = N_{D^{(i)*}}(v)$.*
- 2) *If every vertex $v \in D^{(i)} \setminus \partial_{out}D^{(i)}$ is marked then every vertex $v \in D^{(i)*} \setminus \partial_{out}D^{(i)*}$ is marked.*
- 3) *If every cut vertex in $D^{(i)} \setminus \partial_{in}D^{(i)}$ is either marked or active then every cut vertex in $D^{(i)*} \setminus \partial_{in}D^{(i)*}$ is either marked or active.*
- 4) *Given a diagram D there are only finitely many possible outcomes for D^* .*
- 5) *If $(\partial_{out}D^{(1)} \setminus M(D^{(1)})) \cup (\partial_{out}D^{(2)} \setminus M(D^{(2)})) \neq \emptyset$ then $(\partial_{out}D^{(1)*} \setminus M(D^{(1)*})) \cup (\partial_{out}D^{(2)*} \setminus M(D^{(2)*})) \neq \emptyset$.*

Proof. Follows from the description of B_S . ■

Let D be a diagram. The following notation

$$D^* = B_S^n(D)$$

means that D^* is a result of n applications of B_S to the diagram D .

PROPOSITION 17.7.5. *Let $D = (D^{(1)}, D^{(2)}, M(D^{(1)}), M(D^{(2)}), \emptyset)$, $D' = B_S^n(D) = (E^{(1)}, E^{(2)}, M(E^{(1)}), M(E^{(2)}), \emptyset)$. There exists n_1, n_2 such that $n_1 + n_2 = n$ and*

$$D'' = B_S^{n_1}(D) = (E^{(1)}, D^{(2)}, M(E^{(1)}), M(D^{(2)}), \emptyset),$$

$$(D^{(1)}, D^{(2)}, M(D^{(1)}), M(D^{(2)}), \emptyset) = B_S^{n_2}(D'').$$

Proof. The basic extension B_S treats parts $D^{(1)}$ and $D^{(2)}$ of D separately, i.e., changing the part $D^{(i)}$ (where $i = 1, 2$) does not affect the part $D^{(1-i)}$. Therefore, processing a vertex from $D^{(i)}$ and processing a vertex from $D^{(1-i)}$ are commuting operations. ■

REMARK 17.7.6. Let $D^* = B_S^n(D_0)$ where

$$D_0 = (\text{Loop}(w), \text{Loop}(w), \emptyset, \emptyset, \emptyset).$$

The following hold:

- 1) If $s_2 = 1$ then $\overline{D^*}$ is a “forest on a loop”.
- 2) If $s_1 = 1$ then $\overline{D^*}$ is a “forest of cells on a loop”, i.e., diagram without free edges and such that the dual graph is a tree with one cycle.
- 3) If $s_1 + s_2 = 1$ then $\overline{D^*}$ is a “forest of cells and free edges on a loop”.

Let \mathcal{W}_S be the random walk on \mathcal{K}_w relative to B_S which starts with probability 1 at $D_0 = \text{Loop}(w)$ and \mathcal{T} be the corresponding transition tree. In the following lemma we collect some basic properties of \mathcal{W}_S . By D_∞ we denote an infinite path in \mathcal{T} which defines a sequence of extended annular diagrams from \mathcal{K}_w :

$$D_0 = D_{i_0} \rightarrow D_{i_1} \rightarrow \dots \rightarrow D_{i_k} \rightarrow \dots$$

LEMMA 17.7.7 (Properties of \mathcal{T}). *Let $D_{i_j} = (D_{i_j}^{(1)}, D_{i_j}^{(2)}, M(D_{i_j}^{(1)}), M(D_{i_j}^{(2)}), A_{i_j})$. The following hold:*

- 1) Every vertex $v \in (D_{i_j}^{(k)} - \partial_{\text{out}} D_{i_j}^{(k)})$ ($k = 1, 2$) is marked, i.e.,

$$D_{i_j}^{(k)} - \partial_{\text{out}} D_{i_j}^{(k)} \subseteq M(D_{i_j}^{(k)}).$$

- 2) For every marked vertex $v \in D_{i_j}^{(k)}$ the neighborhood of v does not change in $D_{i_m}^{(k)}$ for $m \geq j$, i.e.,

$$N_{D_{i_m}^{(k)}}(v) = N_{D_{i_j}^{(k)}}(v).$$

- 3) Every unmarked vertex $v \in D_{i_j}^{(k)}$ either stays unmarked in all $D_{i_m}^{(k)}$ for $m \geq j$ or eventually becomes active.
- 4) Every active vertex $v \in D_{i_j}^{(k)}$ either stays active in all $D_{i_m}^{(k)}$ for $m \geq j$ (and in this event the case 4) in the description of B_S does not occur) or eventually it becomes marked.

COROLLARY 17.7.8. *The random basic extension B_S is locally stable at every marked vertex v .*

17.7.2. Completeness of B_S . Let $\langle X; R \rangle$ be a finite symmetrized presentation and $w = w(X)$. In this section we show that the basic extension B_S is \mathcal{L}_w -complete. More precisely, we show that for any annular diagram $D \in \overline{\mathcal{L}_w}$ (satisfying certain simple properties) there exists $n \in \mathbb{N}$ such that $D = \overline{B_S^n(D_0)}$, where

$$D_0 = (\text{Loop}(w), \text{Loop}(w), \emptyset, \emptyset, \emptyset).$$

Let $D \in \mathcal{L}_w$ and π_D be a base loop in D . Assume that D contains no loops of length 1 and there is no cell c such that ∂c is not vertex-simple. We cut D along π_D to obtain two annular diagrams D_{in} and D_{out} . As before, turn D_{in} inside out. Clearly,

$$\begin{aligned} \mu(\partial_{in} D_{in}) &= w, & \mu(\partial_{out} D_{in}) &= \mu(\partial_{in} D), \text{ and} \\ \mu(\partial_{in} D_{out}) &= w, & \mu(\partial_{out} D_{out}) &= \mu(\partial_{out} D). \end{aligned}$$

PROPOSITION 17.7.9. *Let $\langle X; R \rangle$ be a finite symmetrized presentation, D an annular diagram over $\langle X; R \rangle$ which does not contain cycles of length 1 and cells c such that ∂c is not vertex-simple. Assume that none of the probabilities in S is trivial. Then there exist n_1 and n_2 such that*

$$(\text{Loop}(w), D_{out}, \emptyset, M(D_{out}), \emptyset) = B_S^{n_1}(D_0),$$

$$D' = (D_{in}, D_{out}, M(D_{in}), M(D_{out}), \emptyset) = B_S^{n_1+n_2}(D_0)$$

and $D = \overline{D}'$, for some $M(D_{in}) \subseteq V(D_{in})$ and $M(D_{out}) \subseteq V(D_{out})$.

Proof. By Proposition 17.7.5 it is enough to show the existence of n_1 only. Let $\partial_{in} D_{out} = e_1, \dots, e_k$, where $e_i = v_i \xrightarrow{x_i} v_{i+1}$ ($i = 1, \dots, k$) and $v_1 = v_{k+1}$. Construct a van Kampen diagram D' over some presentation $\langle X'; R' \rangle$ (it will be defined later) as follows. First, add a vertex v_0 into the inner hole of D_{out} . Then, for each v_i add an edge $v_0 \xrightarrow{x'_i} v_i$ in such a way to not spoil the planarity of the diagram and add a triangular cell v_0, v_i, v_{i+1} labeled with $x'_i x_i x'^{-1}_{i+1}$ (where each x'_i is a new symbol, i.e., $x'_i \notin X$). See Figure 17.12 for example. The presentation $\langle X'; R' \rangle$ is obtained from $\langle X; R \rangle$ by adding all new symbols x'_i to the set X and all relators corresponding to new cells in D' to the set R' .

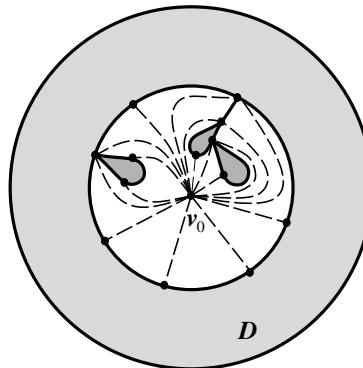


FIGURE 17.12. Annular hole triangulation.

By construction, the diagram D' does not contain loops of length 1 and does not have cells c such that ∂c is not vertex-simple. Hence, D' can be obtained in a

number of applications of B_S for van Kampen diagrams to the trivial diagram D'_0 which consists of exactly one vertex. Moreover, from the proof of the Completeness Theorem for van Kampen diagrams (Chapter 16, Theorem 16.6.6), it follows that we may assume that the only vertex in D'_0 corresponds to the vertex $v_0 \in D'$. Let

$$D'_0 \xrightarrow{B_S} D'_1 \xrightarrow{B_S} \dots \xrightarrow{B_S} D'_k = D'$$

be the corresponding sequence of extended van Kampen diagrams. Let D'_j be the first diagram in which the initial vertex v_0 is marked. Clearly, $\mu(\partial D'_j) = w$ and each vertex in $\partial D'_j$ is unmarked. This is exactly the initial configuration we have in D_0 (except the internal cells, of course). Now, since the basic extension B_S for annular diagrams can perform the same operations as its counterpart for van Kampen diagrams, the proof is done. ■

THEOREM 17.7.10 (Completeness of B_S). *Let $\langle X; R \rangle$ be a finite symmetrized R -reduced presentation and $w = w(X)$ such that w does not belong to the conjugacy class of a word of length less or equal to 1. Let $D \in \mathcal{L}_w$. Then there exists a number n such that*

$$D = \overline{B_S^n(D_0)},$$

i.e., B_S is \mathcal{L}_w complete.

Proof. Observe, that the assumptions on a presentation $\langle X; R \rangle$ and a word w imply that D does not contain loops of length 1. Also, the assumption on $\langle X; R \rangle$ implies that D contains no loops such that ∂c is not vertex-simple. Now, the theorem follows from Proposition 17.7.9. ■

17.8. Asymptotic properties of random annular diagrams

In this section we describe several asymptotic properties of diagrams relative to the basic extension B_S and the complete series of termination conditions $\{\widehat{Q}_i\}_{i \in \mathbb{N}}$. In particular, we discuss asymptotic behavior of the length of the perimeter and the depth of diagrams relative to their size.

All results in this section can be proved by arguments similar to those in Chapter 16 (we leave them to the reader). Let $\langle X; R \rangle$ be a finite symmetrized R -reduced presentation and $w = w(X)$ such that w does not belong to the conjugacy class of a word of length less or equal to 1. Let $\mathcal{L}_{w,n}$ be a set of all annular diagrams from \mathcal{L}_w of size n , i.e.,

$$\mathcal{L}_{w,n} = \{D \in \mathcal{L}_w \mid \chi(D) = n\}.$$

Let $\mathcal{K}_{w,n}$ be a set of all extended annular diagrams from \mathcal{K}_w of size n , i.e.,

$$\mathcal{K}_{w,n} = \{D \in \mathcal{K}_w \mid \chi(D) = n\}.$$

Clearly, the sets $\{\mathcal{L}_{w,n}\}$ is a partition of \mathcal{L}_w and the sets $\{\mathcal{K}_{w,n}\}$ is a partition of \mathcal{K}_w .

THEOREM 17.8.1. *Let $0 < \alpha < 1$. Then the following holds:*

- 1) *Let $\mathcal{K}'_{w,n,\alpha} = \{D \in \mathcal{K}_{w,n} \mid l(D) < (1 - \alpha)n + |w|\} \subseteq \mathcal{K}_{w,n}$. Then $P_{\mathcal{K}_{w,n}}(\mathcal{K}'_{w,n,\alpha}) \rightarrow 0$ exponentially fast as $n \rightarrow \infty$.*
- 2) *Let $\mathcal{L}'_{w,n,\alpha} = \{D \in \mathcal{L}_{w,n} \mid l(D) < (1 - \alpha)n + |w|\} \subseteq \mathcal{L}_{w,n}$. Then $P_{\mathcal{L}_{w,n}}(\mathcal{L}'_{w,n,\alpha}) \rightarrow 0$ exponentially fast as $n \rightarrow \infty$.*

COROLLARY 17.8.2. Let $\langle X; R \rangle$ be a finite symmetrized presentation and $w = w(X)$ such that w does not belong to the conjugacy class of a word of length less or equal to 1. Put

$$\mathcal{L}'_w = \{D \mid l(D) \geq \frac{1}{2}\chi(D)\}.$$

Then

$$\rho_{\mathcal{L}_w}(\mathcal{L}'_w) = \lim_{i \rightarrow \infty} P_{\mathcal{L}_{w,i}}(\mathcal{L}_{w,i} \cap \mathcal{L}'_w) = 1.$$

Moreover, $P_{\mathcal{L}_{w,i}}(\mathcal{L}_{w,i} \cap \mathcal{L}'_w) \rightarrow 1$ exponentially fast. Thus the set of all diagrams over $\langle X; R \rangle$ with linear isoperimetric function (with coefficient $\frac{1}{2}$) is strongly generic with respect to the asymptotic density.

THEOREM 17.8.3. The following holds:

- 1) Let $\mathcal{K}''_{w,n} = \{D \in \mathcal{K}_{w,n} \mid \delta(D) < 2 \log n\} \subseteq \mathcal{K}_{w,n}$. Then $P_{\mathcal{K}_{w,n}}(\mathcal{K}''_{n}) \rightarrow 1$ as $n \rightarrow \infty$.
- 2) Let $\mathcal{L}''_{w,n} = \{D \in \mathcal{L}_{w,n} \mid \delta(D) < 2 \log n\} \subseteq \mathcal{L}_{w,n}$. Then $P_{\mathcal{L}_{w,n}}(\mathcal{L}''_{w,n}) \rightarrow 1$ as $n \rightarrow \infty$.

THEOREM 17.8.4. Let $\langle X; R \rangle$ be a symmetrized R -reduced presentation, $w = w(X)$ such that w does not belong to the conjugacy class of a word of length less or equal to 1, and $0 < \alpha < 1$. The following holds:

- 1) Let $\mathcal{K}'''_{w,n,\alpha} = \{D \in \mathcal{K}_{w,n} \mid \delta(D) < 2 \log n \text{ & } l(D) \geq (1 - \alpha)n\} \subseteq \mathcal{K}_{w,n}$. Then $P_{\mathcal{K}_{w,n}}(\mathcal{K}'''_{w,n,\alpha}) \rightarrow 1$ as $n \rightarrow \infty$.
- 2) Let $\mathcal{L}'''_{w,n,\alpha} = \{D \in \mathcal{L}_{w,n} \mid \delta(D) < 2 \log n \text{ & } l(D) \geq (1 - \alpha)n\} \subseteq \mathcal{L}_{w,n}$. Then $P_{\mathcal{L}_{w,n}}(\mathcal{L}'''_{w,n,\alpha}) \rightarrow 1$ as $n \rightarrow \infty$.

COROLLARY 17.8.5. Let $\langle X; R \rangle$ be a symmetrized R -reduced presentation, $w = w(X)$ such that w does not belong to the conjugacy class of a word of length less or equal to 1. Let

$$\mathcal{L}''_w = \{D \in \mathcal{L}_w \mid \delta(D) \leq 2 \log(2l(D))\}.$$

Then

$$\rho_{\mathcal{L}_w}(\mathcal{L}''_w) = \lim_{i \rightarrow \infty} P_{\mathcal{L}_{w,i}}(\mathcal{L}_{w,i} \cap \mathcal{L}''_w) = 1.$$

Proof. Follows from Theorem 17.8.4 when $\alpha = \frac{1}{2}$. ■

17.9. Asymptotic properties of conjugated words

17.9.1. Random conjugated words. Let $\langle X; R \rangle$ be a finite symmetrized R -reduced presentation and $w = w(X)$ be a word such that the shortest element in its conjugacy class is of length at least 2. Denote by $CP_w(X; R)$ the set of all pairs of cyclic words (w_1, w_2) such that $w_1 \sim_G w \sim_G w_2$. In this section we define a discrete probability measure on $CP_w(X; R)$.

Recall that $\mathcal{L}_{w,i}$ is a set of all annular diagrams from \mathcal{L}_w of size i and $P_{\mathcal{L}_{w,i}}$ is a discrete probability measure on $\mathcal{L}_{w,i}$. Denote by $CW_{w,i}$ (for each $i \in \mathbb{N}$) the set of boundary labels (as cyclic words) of diagrams from $\mathcal{L}_{w,i}$ and by $\overline{CW}_{w,n}$ the union

$$\overline{CW}_{w,n} = \bigcup_{i=1}^n CW_{w,i}.$$

It follows from Proposition 17.5.1 that

$$CP_w(X; R) = \bigcup_{i=1}^{\infty} CW_{w,i} = \bigcup_{i=1}^{\infty} \overline{CW}_{w,i}.$$

Clearly,

$$\overline{CW}_{w,1} \subseteq \overline{CW}_{w,2} \subseteq \overline{CW}_{w,3} \subseteq \dots$$

One can induce probability measures from $(\mathcal{L}_{w,n}, P_{\mathcal{L}_{w,n}})$ onto the sets $CW_{w,n}$ and $\overline{CW}_{w,n}$ as follows. For $S \subseteq CW_{w,n}$ and $S' \subseteq \overline{CW}_{w,n}$ put

$$P_{CW_{w,n}}(S) = P_{\mathcal{L}_{w,n}}(\{D \in \mathcal{L}_{w,n} \mid \text{the boundary labels of } D \text{ belong to } S\}),$$

$$P_{\overline{CW}_{w,n}}(S') = \frac{\sum_{i=1}^n P_{CW_{w,i}}(S' \cap CW_{w,i})}{n}.$$

It is easy to check that $P_{CW_{w,n}}$ and $P_{\overline{CW}_{w,n}}$ are discrete probability measures on $CW_{w,n}$ and $\overline{CW}_{w,n}$.

Using the probability on the sets $\overline{CW}_{w,n}$ one can define an asymptotic density of the subsets of $CP_w(X; R)$ as follows. For $S \subseteq CP_w(X; R)$ put

$$\rho_{CP_w}(S) = \lim_{n \rightarrow \infty} P_{\overline{CW}_{w,n}}(S \cap \overline{CW}_{w,n}).$$

One can describe the probability measure $P_{CW_{w,n}}$ in terms of the following random generator.

ALGORITHM 17.9.1 (Random Generator I of conjugate words).

INPUT. A word $w = w(X)$ and a number $n \in \mathbb{N}$.

OUTPUT. A pair of words w_1, w_2 such that $w_1 \simeq_G w \simeq_G w_2$.

INITIALIZATION. Put $D_0 = (\text{Loop}(w), \text{Loop}(w), \emptyset, \emptyset, \emptyset)$ and $i = 0$.

COMPUTATIONS.

- 1) Let $D_{i+1} = B_S(D_i)$.
- 2) If $\chi(D_{i+1}) \leq n$ then increment i and goto 1).
- 3) Output labels of $\partial_{out} D_i^{(1)}$ and $\partial_{out} D_i^{(2)}$.

The probability measure $P_{\overline{CW}_n}$ can be described in terms of the following random generator.

ALGORITHM 17.9.2 (Random Generator II of conjugate words).

INPUT. A word $w = w(X)$ and a number $n \in \mathbb{N}$.

OUTPUT. A pair of words w_1, w_2 such that $w_1 \simeq_G w \simeq_G w_2$.

COMPUTATIONS.

- 1) Generate randomly and uniformly a number i from the set $\{1, \dots, n\}$.
- 2) Run the defined above generator to generate a random extended diagram D of size i .
- 3) Output the boundary labels of \overline{D} .

In view of the random generators above the probability measures $P_{CW_{w,n}}$ and $P_{\overline{CW}_{w,n}}$ are very natural.

17.9.2. Generic properties of random conjugated words. In this section we study generic properties of conjugated words in $G = \langle X; R \rangle$. Fix α such that $0 < \alpha < 1$ and define

$$\overline{CW}_{w,n,\alpha} = \{(w_1, w_2) \in \overline{CW}_{w,n} \mid w_1 \text{ and } w_2 \text{ are boundary labels}$$

$$\text{of some } D \in \bigcup_{i=1}^n \mathcal{L}_{w,i,\alpha}'''\}.$$

THEOREM 17.9.3. *Let $G = \langle X; R \rangle$ be a finite symmetrized R -reduced presentation and $w = w(X)$ be a word the conjugacy class of which does not contain words of length less than 2. The following holds:*

$$P_{\overline{CW}_{w,n}}(\overline{CW}_{w,n,\alpha}) \rightarrow 1 \text{ as } n \rightarrow \infty.$$

Proof. Follows from the Theorem 17.8.4 and the definition of $P_{\overline{CW}_{w,n}}$. ■

For a fixed $0 < \alpha < 1$ define a set $CP_w^{(\alpha)}$ to be

$$CP_w^{(\alpha)} = \bigcup_{n=1}^{\infty} \overline{CW}_{w,n,\alpha}.$$

By Theorem 17.9.3 we have $\rho_{CP_w}(CP_w^{(\alpha)}) = 1$.

THEOREM 17.9.4. *Let $\langle X; R \rangle$ be a finite symmetrized R -reduced presentation, $G = \langle X; R \rangle$, and $w = w(X)$ be a word the conjugacy class of which does not contain words of length less than 2. Then the time complexity function for Algorithm 17.6.11 (the search algorithm for the conjugacy problem in G) on the set of inputs $(w_1, w_2) \in CP_w^{(\alpha)}$ is bounded from above by the polynomial*

$$O((|w_1| + |w_2|)^{2+4\log L(R)}).$$

Proof. Follows from Theorem 17.6.12 and the definition of the set $CP_w^{(\alpha)}$. ■

Thus, the conjugacy search problem is polynomial on a generic subset $CP_w^{(\alpha)}$ of instances of the problem CP_w .

Part 6

Word Problem in some Special Classes of Groups

CHAPTER 18

Free Solvable Groups

In this chapter we study the computational complexity of several algorithmic problems related to the word problem (WP) in free solvable groups. Let $S_{r,d}$ be a free solvable group of rank $r \geq 2$ and the solvability class $d \geq 2$. We present here a uniform decision algorithm that solves WP in time $O(rn \log_2 n)$ in the free metabelian group $S_{r,2}$ (also denoted by M_r), and $O(n^3rd)$ in the free solvable group $S_{r,d}$ for $d \geq 3$, where n is the length of the input word. In particular, this algorithm is at most cubic in n and linear in r and d for all free solvable groups $S_{r,d}$. Notice, that in all previously known polynomial time decision algorithms for WP in $S_{r,d}$ the degree of the polynomial grows together with d . In fact, we prove more. We show that one can compute Fox derivatives of elements from $S_{r,d}$ in time $O(n^3rd)$. This allows one to use efficiently the Magnus embedding in computations with free solvable groups. On the other hand, we describe geodesics in $S_{r,d}$ and show that a seemingly close problem of finding the geodesic length of a given element from $S_{r,2}$ is surprisingly hard; it is **NP**-complete. Our approach is based on such classical tools as the Magnus embedding and Fox calculus, as well as, on relatively new (in group theory) geometric ideas from [62] and [278]. In particular, we establish a direct link between Fox derivatives and geometric flows on Cayley graphs.

The study of algorithmic problems in free solvable groups can be traced to the work of Magnus [178], who in 1939 introduced an embedding (now called the Magnus embedding) of an arbitrary group of the type F/N' into a matrix group of a particular type with coefficients in the group ring of F/N (see section 18.1.2 below). Since WP in free abelian groups is decidable in polynomial time, by induction, this embedding immediately gives a polynomial time decision algorithm for a fixed free solvable group $S_{r,d}$. However, the degree of the polynomial here grows together with d .

In 1950s R. Fox introduced his free differential calculus and made the Magnus embedding much more transparent [78, 79, 80, 38] (see also Section 18.1.3). Namely, besides other things, he showed that an element $w \in F$ belongs to $N' = [N, N]$ if and only if all partial derivatives of w are equal to 0 in the integer group ring of F/N . This reduces WP in F/N' directly to the word problem in F/N . In particular, it solves, by induction, WP in $S_{r,d}$. Again, the decision algorithm is polynomial in a fixed group $S_{r,d}$, but the degree of the polynomial grows with d ; not a surprise since the partial derivatives of w describe precisely the image of w under the Magnus embedding.

A few years later P. Hall proved the finiteness of all subdirect indecomposable finitely generated abelian-by-nilpotent groups. This implies that all finitely generated abelian-by-nilpotent, in particular, metabelian, groups are residually finite. About the same time Gruenberg extended this result to arbitrary free solvable groups [121]. Now one can solve WP in $S_{r,d}$ in the following way. Given $w \in S_{r,d}$,

as a word in the fixed set of generators, one can start two processes in parallel. The first one enumerates effectively all consequences of the defining relations of $S_{r,d}$ in F_r (which is possible since the group is recursively presented) until the word w occurs, and the second one enumerates all homomorphisms from $S_{r,d}$ into all finite symmetric groups S_n (checking if a given r -tuple of elements in S_n generates a solvable group of class d) until it finds one where the image of w is non-trivial. However, computer experiments show that the algorithm described above is extremely inefficient (though its complexity is unknown).

Another shot at WP in metabelian groups comes from their linear representations. V. Remeslennikov proved in [231] that a finitely generated metabelian group (under some restrictions) is embeddable into $GL(n, R)$ for a suitable n and a suitable ring $R = K_1 \times \dots \times K_n$ which is a finite direct product of fields K_i . In [281], see also [282], B. Wehrfritz generalized this result to arbitrary finitely generated metabelian groups G . It follows that G is embeddable into a finite direct product of linear groups. Since WP in linear groups is polynomial time decidable this implies that WP in G is polynomial time decidable. Notice, that it is unclear if there is a uniform polynomial time decision algorithm for WP in arbitrary finitely generated metabelian groups.

By comparison, observe that there are finitely presented solvable groups of class 3 with undecidable WP. In [157] O. Kharlampovich constructed the first example of such a group by reducing the halting problem for the universal Minski machine to WP of the group. There are several results which clarify the boundary between decidability and undecidability of the word problems in solvable groups, we refer to a survey [158] for details.

Our approach to WP in free solvable groups is based on the Fox theorem mentioned above. Using binary tree search techniques and associative arrays we are able to compute Fox's derivatives of elements w of a free solvable group $S_{r,d}$ in time $O(n^3d)$, where $n = |w|$. Significance of this result goes beyond WP for these groups; it gives a fast algorithm to compute images of elements under the Magnus embedding. This opens up an opportunity to solve effectively other algorithmic problems in groups $S_{r,d}$ using the classical techniques developed for wreath products of groups.

In Section 18.3 we study algorithmic problems on geodesics in free metabelian groups. Let G be a group with a finite set of generators $X = \{x_1, \dots, x_r\}$ and let $\mu : F(X) \rightarrow G$ be the canonical epimorphism. For a word w in the alphabet $X^{\pm 1}$ by $|w|$ we denote the *length* of w . The *geodesic length* $l_X(g)$ of an element $g \in G$ relative to X is defined by

$$l_X(g) = \min\{|w| \mid w \in F(X), w^\mu = g\}.$$

We sometimes write $l_X(w)$ instead of $l_X(w^\mu)$. A word $w \in F(X)$ is called *geodesic* in G relative to X , if $|w| = l_X(w)$. We are interested here in the following two algorithmic *search* problems in G .

The Geodesic Problem (GP): Given a word $w \in F(X)$ find a word $u \in F(X)$ which is geodesic in G such that $w^\mu = u^\mu$.

The Geodesic Length Problem (GLP): Given a word $w \in F(X)$ find $l_X(w)$.

One can stratify the search problem GLP to get the corresponding bounded *decision* problem:

The Bounded Geodesic Length Problem (BGLP): Let G be a group with a finite generating set X . Given a word $w \in F(X)$ and a natural number k determine if $l_X(w) \leq k$.

These three problems are polynomial-time reducible to each other. Among general facts on computational complexity of geodesics notice that if G has a polynomial *growth*, i.e., there is a polynomial $p(n)$ such that for each $n \in \mathbb{N}$ cardinality of the ball B_n of radius n in the Cayley graph $\Gamma(G, X)$ is at most $p(n)$, then one can easily construct this ball B_n in polynomial time with an oracle for WP in G . If, in addition, such a group G has WP decidable in polynomial time, then all the problems above have polynomial time complexity with respect to any finite generating set of G (since the growth and WP stay polynomial for any finite set of generators). Now, by Gromov's theorem [117], groups of polynomial growth are virtually nilpotent, hence linear, so they have WP decidable in polynomial time. It follows that all geodesic problems are polynomial time decidable in groups of polynomial growth (finitely generated virtually nilpotent groups). On the other hand, there are many groups of exponential growth where GP is decidable in polynomial time, for example, hyperbolic groups [69] or metabelian Baumslag-Solitar group $BS(1, n) = \langle a, t \mid t^{-1}at = a^n \rangle$, $n \geq 2$ (see [66] and Section 18.3.1 for comments).

In general, if WP in G is decidable in polynomial time then BGLP is in the class **NP**, i.e., it is decidable in polynomial time by a non-deterministic Turing machine. It might happen though, that BGLP in a group G is **NP**-complete. The simplest example of this type is due to Perry, who showed in [220] that BGLP is **NP**-complete in the metabelian group $\mathbb{Z}_2 wr (\mathbb{Z} \times \mathbb{Z})$ (the wreath product of \mathbb{Z}_2 and $\mathbb{Z} \times \mathbb{Z}$). Therefore, the problems GP and GLP are **NP**-complete too.

Our viewpoint on geodesics in free solvable groups is based on geometric ideas from the following two papers. In 1993 Droms, Lewin, and Servatius introduced a new geometric approach to study WP and GLP in groups of the type F/N' via paths in the Cayley graph of F/N [62]. In 2004 Vershik and Dobrynin studied algebraic structure of solvable groups, using homology of related Cayley graphs [278]. This approach was outlined earlier in the papers [276, 277], where possible applications to random walks on metabelian groups have been discussed. In the papers [276, 277] (see also [278]) a new robust presentation of a free metabelian group $S_{r,2}$ was introduced as an extension of \mathbb{Z}^r by the integer first homology group of the lattice \mathbb{Z}^r (viewed as a one-dimensional complex) with a distinguished 2-cocycle. Similar presentations of other metabelian and solvable groups laid out foundations of a new approach to algorithmic problems in solvable groups.

It seems these ideas are still underdeveloped in group-theoretic context, despite their obvious potential. Meanwhile, in semigroup theory similar geometric techniques have been widely used to deal with free objects in semidirect products of varieties. One can find an explicit exposition of these techniques in the papers due to Almeida [3] and Almeida and Weil [4], while in [234, 10, 11, 9] Aninger, Rhodes and Steinberg use similar machinery on a regular basis. Earlier, similar methods, though sometimes implicitly, were used in inverse semigroups theory, we refer here to papers [208, 182, 183, 46].

In group theory most of the results in this area relied on various forms of the Magnus embedding and Fox derivatives. The role that the Magnus embeddings play in varieties of groups was clarified by Shmelkin [245]. In [187] Matthews proved that the conjugacy problem (CP) in free metabelian groups is decidable,

and Kargapolov and Remeslennikov generalized this to free solvable groups $S_{r,d}$ [150]. A few years later Remeslennikov and Sokolov described precisely the image of F/N' under the Magnus embedding and showed that CP is residually finite in $S_{r,d}$ [233]. We refer to a survey [232] on algorithmic problems in solvable groups.

In Sections 18.1.4 and 18.1.5 we study elements of groups of the type F/N' via flows on the Cayley graph Γ of F/N . It turns out the flow generated by a word $w \in F$ on the graph Γ directly corresponds to the Fox derivatives of w in the group ring $\mathbb{Z}F/N$. This simple observation links together the techniques developed in group theory for the Magnus embeddings with the extensive geometric and the graph-theoretic machinery for flows. Indeed, the set of geometric circulations (flows where the Kirchhoff law holds for all vertices, including the source and the sink) forms a group which is naturally isomorphic to the first homology group $H_1(\Gamma, \mathbb{Z})$ of Γ . In this context the geometric circulations on Γ represent precisely the 1-cycles of Γ (viewed as 1-complex). The classical result in homology theory describes $H_1(\Gamma, \mathbb{Z})$ as the abelianization of the fundamental group $\pi_1(\Gamma)$, which, in this case, is isomorphic to the free group N . Putting all these together one has another geometric proof of the Fox theorem, as well as the description of the kernel of the Magnus embedding.

In Section 18.1.7 we describe geodesics in groups F/N' as Euler tours in some finite subgraphs of Γ generated by the supports of the flows of the elements of F/N' on Γ . The description is geometric, explicit, and it gives a natural way to compute the geodesic length of elements. In this part geometric ideas seem unavoidable. However, this simplicity becomes treacherous when one concerns the efficiency of computations.

We prove that BGLP (relative to the standard basis) is **NP**-complete even in $S_{r,2}$. Therefore, the problems GP and GLP are **NP**-complete in $S_{r,2}$ too. To show this we construct a polynomial-time reduction of the Rectilinear Steiner tree problem (RSTP), which is **NP**-complete, to BGLP in $S_{r,2}$. The necessary information on RSTP is outlined in Section 18.3.3 and the proof of the main theorem is in Section 18.3.4. Notice, that in [62] GLP was claimed to be polynomial time decidable in arbitrary finitely generated free solvable groups, but the argument turned out to be fallacious.

In the second half of the 20th century free solvable groups, as well as, solvable wreath products of groups and finitely generated metabelian groups, were intensely studied, but mostly from the view-point of combinatorial groups theory. Now they stand at the heart of research in various areas of algebra. On the one hand, the rejuvenated interest to these groups stems from random walks on groups and, cohomology theory. For example, wreath products of abelian groups give exciting examples and counterexamples to several conjectures on the numerical characteristics of random walks. It seems, the main reasons that facilitate research here come from some paradoxical properties of the groups itself: all these groups are amenable (as solvable group), but they have exponential growth and may have nontrivial Poisson boundary [143], etc. These groups, contrary to, say, free nilpotent groups, may have irreducible unitary representations with nontrivial cohomology. Some numerical characteristics of these groups are very intriguing, giving new exciting examples in the quantitative group theory. For example, metabelian “lamplighter” groups have intermediate growth of the drift, positive entropy, etc. These groups

were intensively studied recently (see papers [143, 70] and the bibliography in the latter).

On the other hand, metabelian groups are currently at the focus of a very active research in geometric groups theory. In 1983 Gromov proposed a program for studying finitely generated groups as geometric objects [118]. One of the principal directions of this program is the classification of finitely generated groups up to quasi-isometry. It follows from Gromov's result on groups with polynomial growth [117] that a group quasi-isometric to a nilpotent group is virtually nilpotent. In the case of solvable groups the situation is much less known. Erschler showed in [64] that a group quasi-isometric to a solvable group may be not virtually solvable. Thus, the class of virtually solvable groups is not closed under quasi-isometry. On the other hand, there are interesting classes of solvable non-polycyclic groups that are quasi-isometrically rigid, for example, solvable Baumslag-Solitar groups (Farb and Mosher [72, 73]). We refer to the papers [203] and [71] for some recent results in this area.

18.1. Preliminaries

18.1.1. The word problem. Let $F = F_r = F(X)$ be a free group with a basis $X = \{x_1, \dots, x_r\}$. A subset $R \subseteq F$ defines a *presentation* $P = \langle X \mid R \rangle$ of a group $G = F/N$ where $N = ncl(R)$ is the *normal closure* of R in F . If R is finite (recursively enumerable) then the presentation is called finite (recursively enumerable).

We say that the word problem (WP) for P is *decidable* if the normal closure N is a decidable subset of $F(X)$, i.e., there exists an algorithm \mathcal{A} to decide whether a given word $w \in F(X)$ belongs to N or not. The *time function* $T_{\mathcal{A}} : F(X) \rightarrow \mathbb{N}$ of the algorithm \mathcal{A} is defined as the number of steps required for \mathcal{A} to halt on an input $w \in F(X)$. We say that the word problem for P is decidable in polynomial time if there exists a decision algorithm \mathcal{A} , as above, and constants $c, k \in \mathbb{N}$ such that

$$T_{\mathcal{A}}(w) \leq c|w|^k$$

for every $w \in F(X)$ (here $|w|$ is the length of the word w). In this case we say that the time complexity of WP for P is $O(n^k)$.

18.1.2. Free solvable groups and the Magnus embedding. For a free group $F = F(X)$ of rank r denote by $F^{(1)} = F' = [F, F]$ the *derived subgroup* of F , and by $F^{(d)} = [F^{(d-1)}, F^{(d-1)}]$ – the d -th *derived subgroup* of F , $d \geq 2$. The quotient group $A_r = F_r/F'_r$ is a *free abelian group* of rank r , $M_r = F_r/F_r^{(2)}$ is a *free metabelian group* of rank r , and $S_{r,d} = F_r/F_r^{(d)}$ is a *free solvable group* of rank r and class d . In the sequel we usually identify the set X with its canonical images in A_r , M_r and $S_{r,d}$.

One of the most powerful approaches to study free solvable groups is via the Magnus embedding. To explain we need to introduce some notation. Let $G = F/N$ and $\mathbb{Z}G$ the group ring of G with integer coefficients. By $\mu : F \rightarrow G$ we denote the canonical factorization epimorphism, as well its linear extension to $\mu : \mathbb{Z}F \rightarrow \mathbb{Z}G$. Let T be a free (left) $\mathbb{Z}G$ -module of rank r with a basis $\{t_1, \dots, t_r\}$. Then the set of matrices

$$M(G) = \left(\begin{array}{cc} G & T \\ 0 & 1 \end{array} \right) = \left\{ \left(\begin{array}{cc} g & t \\ 0 & 1 \end{array} \right) \mid g \in G, t \in T \right\}$$

forms a group with respect to the matrix multiplication. It is easy to see that the group $M(G)$ is a discrete wreath product $M(G) = A_r wr G$ of the free abelian group A_r and the group G .

THEOREM 18.1.1 ([178]). *The homomorphism $\varphi : F \rightarrow M(G)$ defined by*

$$x_i \mapsto \begin{pmatrix} x_i^\mu & t_i \\ 0 & 1 \end{pmatrix}, \quad i = 1, \dots, r,$$

satisfies $\ker \varphi = N'$. Therefore, φ induces a monomorphism

$$\varphi : F/N' \hookrightarrow M(F/N).$$

The monomorphism φ is now called the *Magnus embedding*. The Magnus embedding allows one to solve WP in the group F/N' if WP in $G = F/N$ is decidable. Indeed, observe that

$$x_i^{-1} \mapsto \begin{pmatrix} (x_i^{-1})^\mu & (-x_i^{-1})^\mu t_i \\ 0 & 1 \end{pmatrix}, \quad i = 1, \dots, r.$$

Now, if for $i = 1, \dots, r$ and $\varepsilon = \pm 1$ we define the value

$$\delta(x_i^\varepsilon) = \begin{cases} 1, & \text{if } \varepsilon = 1; \\ (-x_i^{-1})^\mu, & \text{if } \varepsilon = -1; \end{cases}$$

then given a word $w = x_{i_1}^{\varepsilon_1} \dots x_{i_n}^{\varepsilon_n} \in F(X)$ one can compute its image $\varphi(w)$ in $M(G)$ as follows

$$\begin{aligned} \varphi(w) &= \varphi(x_{i_1}^{\varepsilon_1}) \dots \varphi(x_{i_n}^{\varepsilon_n}) \\ &= \begin{pmatrix} \mu(x_{i_1}^{\varepsilon_1}) & \delta(x_{i_1}^{\varepsilon_1})t_{i_1} \\ 0 & 1 \end{pmatrix} \cdot \dots \cdot \begin{pmatrix} \mu(x_{i_n}^{\varepsilon_n}) & \delta(x_{i_n}^{\varepsilon_n})t_{i_n} \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \mu(x_{i_1}^{\varepsilon_1} \dots x_{i_n}^{\varepsilon_n}) & \sum_{j=1}^n (x_{i_1}^{\varepsilon_1} \dots x_{i_{j-1}}^{\varepsilon_{j-1}})^\mu \delta(x_{i_j}^{\varepsilon_j})t_{i_j} \\ 0 & 1 \end{pmatrix} \end{aligned}$$

and then, using a decision algorithm for WP in G , check if the resulting matrix $\varphi(w)$ is the identity matrix or not. To estimate the complexity of such an algorithm notice first, that the coefficients from $\mathbb{Z}G$ that occur in the upper-right corner of the matrix $\varphi(w)$ have $O(|w|)$ summands. Second, to check whether or not an element $h = m_1 v_1 + \dots + m_k v_k \in \mathbb{Z}G$, where $m_i \in \mathbb{Z}$ and $v_i \in G$ are given as words in the generators X from G , is trivial in $\mathbb{Z}G$ it requires $O(k^2)$ comparisons of the type $v_i = v_j?$ in G . This gives an estimate for the time function T' of WP in F/N' via the time function T for WP in F/N :

$$T'(n) = O(rn^2 T(n)),$$

where $n = |w|$. Since WP in A_r can be decided in linear time the estimate above shows that the complexity of WP in M_r is $O(rn^3)$. Moreover, induction on the solvability class d gives a polynomial estimate $O(r^{d-1} n^{2d-1})$ for WP in the free solvable group $S_{r,d}$. Thus, the Magnus embedding gives a straightforward polynomial time (in r and n) decision algorithm for WP in $S_{r,d}$, but the degree of the polynomial grows with d . In particular, this algorithm is not polynomial as a uniform algorithm on the whole class of free solvable groups.

18.1.3. Free Fox derivatives. Let $F = F_r(X)$ be a free group of rank r with a basis $X = \{x_1, \dots, x_r\}$. The trivial group homomorphism $F \rightarrow 1$ extends to a ring homomorphism $\varepsilon : \mathbb{Z}F \rightarrow \mathbb{Z}1 \simeq \mathbb{Z}$. The kernel of ε is called *the fundamental ideal* Δ_F of $\mathbb{Z}F$, it is a free (left) $\mathbb{Z}F$ -module freely generated by elements $x_1 - 1, \dots, x_r - 1$.

In [78, 79, 80, 38] R. Fox introduced and gave a thorough account of the free differential calculus in the group ring $\mathbb{Z}F$. Here we recall some notions and results referring to the books [48, 21, 122] for details.

A map $D : \mathbb{Z}F \rightarrow \mathbb{Z}F$ is called a *derivation* if it satisfies the following conditions:

$$(D1) \quad D(u + v) = D(u) + D(v);$$

$$(D2) \quad D(uv) = D(u)v^\varepsilon + uD(v), \text{ where } \varepsilon \text{ is the ring homomorphism defined above.}$$

For every $x_i \in X$ there is a unique derivation, the so-called, free partial derivative $\partial/\partial x_i$, such that $\partial x_j / \partial x_i = \delta_{ij}$, where δ_{ij} is the Kronecker's delta. It turned out that for every $u \in \mathbb{Z}F$,

$$(63) \quad u - u^\varepsilon = \frac{\partial u}{\partial x_1}(x_1 - 1) + \dots + \frac{\partial u}{\partial x_r}(x_r - 1).$$

Since Δ_F is a free $\mathbb{Z}F$ -module the equality (63) gives another definition of the partial derivatives.

Condition (D2) implies the following useful formulas, that allow one to compute easily partial derivatives of elements of $\mathbb{Z}F$,

$$(64) \quad \frac{\partial}{\partial x_i}(uv) = \frac{\partial}{\partial x_i}u + u\frac{\partial}{\partial x_i}v, \quad \text{for any } u, v \in F$$

and

$$(65) \quad \frac{\partial}{\partial x_i}(u^{-1}) = -u^{-1}\frac{\partial}{\partial x_i}u, \quad \text{for any } u \in F.$$

Therefore,

$$(66) \quad \frac{\partial}{\partial x_i}(x_j^{-1}) = -\delta_{ij}x_j^{-1}.$$

and, hence, for a word $w = x_{i_1}^{\varepsilon_1} \dots x_{i_n}^{\varepsilon_n} \in F(X)$ one has

$$(67) \quad \begin{aligned} \frac{\partial w}{\partial x_i} &= \sum_{j=1}^n x_{i_1}^{\varepsilon_1} \dots x_{i_{j-1}}^{\varepsilon_{j-1}} (\partial x_{i_j}^{\varepsilon_j} / \partial x_i) \\ &= \sum_{1 \leq j \leq n, i_j=i, \varepsilon_j=1} x_{i_1}^{\varepsilon_1} \dots x_{i_{j-1}}^{\varepsilon_{j-1}} - \sum_{1 \leq j \leq n, i_j=i, \varepsilon_j=-1} x_{i_1}^{\varepsilon_1} \dots x_{i_j}^{\varepsilon_j}. \end{aligned}$$

The following result is one of the principle technical tools in this area, it follows easily from the Magnus embedding theorem, but in the current form it is due to Fox [78, 79, 80, 38].

THEOREM (Fox). *Let N be a normal subgroup of F and $\mu : F \rightarrow F/N$ the canonical epimorphism. Then for every $u \in F$ the following equivalence holds:*

$$\forall i \quad (\partial u / \partial x_i)^\mu = 0 \iff u \in [N, N].$$

In particular, for $N = F^{(d)}$ the standard epimorphism $\mu : F \rightarrow S_d = F/F^{(d)}$ gives rise to a ring homomorphism $\mu : \mathbf{Z}F \rightarrow \mathbf{Z}S_d$ such that

$$(68) \quad F^{(d+1)} = \{u \in F \mid (\partial u / \partial x_i)^\mu = 0 \text{ for } i = 1, \dots, r\}.$$

Composition of $\partial / \partial x_i$ with μ gives an *induced* partial derivative $\partial^\mu / \partial x_i : \mathbf{Z}F \rightarrow \mathbf{Z}S_d$, which we often denote again by $\partial / \partial x_i$ omitting μ (when it is clear from the context).

Partial derivatives $\partial^\mu / \partial x_i$ are useful when computing images under the Magnus embedding. Indeed, by induction on the length of $w \in F$ it is easy to show that the image of w under the Magnus embedding $\varphi : F/N' \rightarrow M(F/N)$ can be written as follows:

$$w^\varphi = \begin{pmatrix} w^\mu & \sum_{i=1}^r \partial^\mu w / \partial x_i \cdot t_i \\ 0 & 1 \end{pmatrix}.$$

This shows that the faithfulness of the Magnus embedding is, in fact, equivalent to the Fox theorem above.

18.1.4. Flows on F/N . In this section we relate flow networks on the Cayley graph of F/N to the elements of F/N' .

Let $X = \{x_1, \dots, x_r\}$ be a finite alphabet. An *X-labeled directed graph* Γ (or *X-digraph*) is a pair of sets (V, E) where the set V is called the *vertex set* and the set $E \subseteq V \times V \times X$ is called the *edge set*. An element $e = (v_1, v_2, x) \in E$ designates an edge with the *origin* v_1 (also denoted by $o(e)$), the *terminus* v_2 (also denoted by $t(e)$), labeled by x . If for $e \in E$ we have $o(e) = t(e)$ then we say that e is a *loop*. The graph Γ can be finite or infinite.

EXAMPLE 18.1.2. The Cayley graph $\Gamma(G, X)$ of the group $G = F/N$ is an X -digraph.

Given an X -digraph Γ , we can make Γ into a directed graph labeled by the alphabet $X^{\pm 1} = X \cup X^{-1}$. Namely, for each edge $e = (v_1, v_2, x)$ of Γ we introduce a formal inverse $e^{-1} = (v_2, v_1, x^{-1})$. For the new edges e^{-1} we set $(e^{-1})^{-1} = e$. The new graph, endowed with this additional structure, is denoted by $\hat{\Gamma}$. In fact, in many instances we abuse notation by disregarding the difference between Γ and $\hat{\Gamma}$.

REMARK 18.1.3. If X is a generating set of G such that $X \cap X^{-1} = \emptyset$ then $\hat{\Gamma}(G, X)$ is the Cayley graph $\Gamma(G, X^{\pm 1})$ of G relative to the generating set $X^{\pm 1}$.

The edges of $\hat{\Gamma}$ inherited from Γ are called *positively oriented* or *positive*. The formal inverses of positive edges in $\hat{\Gamma}$ are called *negatively oriented* or *negative*. The edge set of $\hat{\Gamma}$ splits in a disjoint union $E(\hat{\Gamma}) = E^+(\Gamma) \sqcup E^-(\Gamma)$ of the sets of positive and negative edges.

The use of $\hat{\Gamma}$ allows us to define the notion of a *path* in Γ . Namely, a *path* p in Γ is a sequence of edges $p = e_1, \dots, e_k$ where each e_i is an edge of $\hat{\Gamma}$ and the origin of each e_i (for $i > 1$) is the terminus of e_{i-1} . In this situation we say that the *origin* $o(p)$ of p is $o(e_1)$ and the *terminus* $t(p)$ is $t(e_k)$. The *length* $|p|$ of this path is set to be k . Also, such a path p has a naturally defined label $\nu(p) = \nu(e_1) \dots \nu(e_k)$. Thus $\nu(p)$ is a word in the alphabet $\Sigma = X \cup X^{-1}$. Note that it is possible that $\nu(p)$ contains subwords of the form aa^{-1} or $a^{-1}a$ for some $a \in X$. If v is a vertex

of Γ , we will consider the sequence $p = v$ to be a path with $o(p) = t(p) = v$, $|p| = 0$ and $\nu(p) = 1$ (the empty word).

In general, one can consider labels in an arbitrary inverse semigroups, the construction above applies to this case as well. In particular, we will consider directed graphs with labels in \mathbb{Z} . We consider also digraphs with no labels at all (to unify terminology, we view them sometimes, as labeled in the trivial semigroup $\{1\}$), the construction above still applies.

Let $\Gamma = (V, E)$ be an X -digraph with two distinguished vertices s (called *source*) and t (called *sink*) from V . Recall that a *flow* (more precisely \mathbb{Z} -flow) on Γ is a function $f : E \rightarrow \mathbb{Z}$ such that

(F) for all $v \in V - \{s, t\}$ the equality $\sum_{o(e)=v} f(e) - \sum_{t(e)=v} f(e) = 0$ holds.
The number $f^*(v) = \sum_{o(e)=v} f(e) - \sum_{t(e)=v} f(e)$ is called the *net flow* at $v \in V$. The condition (F) is often referred to as the *Kirchhoff law* (see, for example, [27, 61]) or the *conservation law* [37].

For the digraph $\hat{\Gamma}$ the definition above can be formulated in the following equivalent way, which is the standard one in flow networks:

- (F1) $f(e) = -f(e^{-1})$ for any $e \in E$.
- (F2) $\sum_{o(e)=v} f(e) = 0$ for all $v \in V - \{s, t\}$.

Here the net flow at v is equal to $f^*(v) = \sum_{o(e)=v} f(e)$.

Usually a flow network comes equipped with a *capacity* function $c : E \rightarrow \mathbb{N}$, in which case a flow f has to satisfy the *capacity restriction*

- (F3) $f(e) \leq c(e)$ for all $e \in E$.

In the sequel we do not make much use of the capacity function (it occurs in an obvious way), so in most cases we consider flows on graphs Γ satisfying the Kirchhoff law (F) (or, equivalently, on graphs $\hat{\Gamma}$ satisfying (F1) and (F2)).

A flow f is called a *circulation* if (F) holds for all vertices from V (including the source s and the sink t).

EXAMPLE 18.1.4. Let $\Gamma = \Gamma(G, X)$ be the Cayley graph of $G = F/N$ relative to the generating set X . The constant function $f : E(\Gamma) \rightarrow \{1\}$ defines a circulation on Γ , since for every vertex $g \in V(\Gamma)$ and every label $x \in X$ there is precisely one edge (gx^{-1}, g) with label x incoming into g and precisely one edge (g, gx) with the label x leaving g .

An important class of flows on $\Gamma = \Gamma(G, X)$ comes from paths in Γ . A path p in Γ defines an integer-valued function $\pi_p : E(\Gamma) \rightarrow \mathbb{Z}$, such that for an edge e $\pi_p(e)$ is the algebraic sum (with respect to the orientation) of the number of times the path p traverses e , i.e., each traversal of e in the positive direction adds $+1$, and in the negative direction adds -1 . It is obvious that π_p is a flow on Γ with the source $o(p)$ (the initial vertex of p) and the sink $t(p)$ (the terminal vertex of p). Notice that π_p satisfies also the following conditions:

- (F4) Either π_p is a circulation (iff p is a closed path), or $f^*(s) = 1, f^*(t) = -1$.
- (F5) π_p has finite support, i.e, the set $\text{supp}(\pi_p) = \{e \in E \mid \pi_p(e) \neq 0\}$ is finite.

We say that a flow π on Γ is *geometric* if it satisfies conditions (F4) and (F5).

It is easy to see that the set $\mathcal{C}(\Gamma)$ of all circulations on Γ forms an abelian group with respect to the operations (here $f, g \in \mathcal{C}(\Gamma)$):

- $(f + g)(e) = f(e) + g(e)$,

- $(-f)(e) = -f(e)$.

Meanwhile, the set $\mathcal{GC}(\Gamma)$ of all geometric circulations is a subgroup of $\mathcal{C}(\Gamma)$. On the other hand, the sum $f + g$ of two geometric flows gives a geometric flow only if the sink of f is equal to the source of g , or either f or g (or both) is a circulation. In fact, the set $\mathcal{GF}(\Gamma)$ of all geometric flows is a groupoid.

Let $\Pi(\Gamma)$ be the fundamental groupoid of paths in Γ . Then the map $\sigma : \Pi(\Gamma) \rightarrow \mathcal{GF}(\Gamma)$ defined for $p \in \Pi(\Gamma)$ by $\sigma(p) = \pi_p$ is a morphism in the category of groupoids, i.e., the following holds (here $p, q \in \Pi(\Gamma)$):

- $\pi_{pq} = \pi_p + \pi_q$, if pq is defined in $\Pi(\Gamma)$,
- $\pi_{p^{-1}} = -\pi_p$.

Now we will show that every geometric flow π on Γ can be realized as a path flow π_p for a suitable path p .

LEMMA 18.1.5. *Let π be a geometric flow on Γ . Then there exists a path p in Γ such that $\pi = \pi_p$.*

Proof. Let π be a geometric flow on Γ with the source s and the sink t . Denote by Γ_π the subgraph of Γ generated by $\text{supp}(\pi) \cup \{s, t\}$. Suppose Q is a subgraph of Γ such that $\Delta = \Gamma_\pi \cup Q$ is a connected graph (every two vertices are connected by a path in $\hat{\Gamma}$). Clearly, π induces a flow on Δ . Now we construct another X -digraph Δ^* by adding new edges to Δ in the following manner. For every edge $e \in E(\Delta)$ with $|\pi(e)| > 1$ we add extra $|\pi(e)| - 1$ new edges $e^{(1)}, \dots, e^{|\pi(e)|-1}$ from $o(e)$ to $t(e)$ if $\pi(e) > 0$ and from $t(e)$ to $o(e)$, if $\pi(e) < 0$. We label the new edges by the same label if $\pi(e) > 0$, and by its inverse, otherwise. If $\pi(e) = 0$ then we add a new edge e^{-1} from $t(e)$ to $o(e)$ with the inverse label. In the case $|\pi(e)| = 1$ we do not add any new edges. Notice that every vertex in $V(\Delta^*) - \{s, t\}$ has even directed degree (the number of incoming edges is equal to the number of outgoing edges). There are two cases to consider.

Case 1). Suppose π is a circulation. Then every vertex in Δ^* has even directed degree. Therefore, the digraph Δ^* has an Euler tour p^* , i.e., a closed path at s that traverses every edge in Δ^* precisely once. Let $\varphi : \Delta^* \rightarrow \hat{\Delta}$ be the morphism of X -digraphs that maps all the new edges $e^{(1)}, \dots, e^{|\pi(e)|-1}$ to their original edge e . Clearly, the image $p = \varphi(p^*)$ is a path in $\hat{\Delta}$ such that $\pi_p = \pi$.

Case 2). Suppose π is not a circulation. Let q be a path in Δ^* from s to t . Then $\pi' = \pi - \pi_q$ is a circulation. Hence by Case 1) there exists a path p in Γ such that $\pi' = \pi_p$. Therefore, $\pi_{pq} = \pi_p + \pi_q = \pi' + \pi_q = \pi$, as required. ■

18.1.5. Geometric interpretation of Fox derivatives. In this section we give a geometric interpretation of Fox derivatives.

Let $G = F/N$, $\mu : F \rightarrow F/N$ the canonical epimorphism, and $\Gamma = \Gamma(G, X)$ the Cayley graph of G with respect to the generating set X . A word $w \in F(X)$ determines a unique path p_w in Γ labeled by w which starts at 1 (the vertex corresponding to the identity of G). As we mentioned in Section 18.1.4 the path p_w defines a geometric flow π_{p_w} on Γ which we denote by π_w .

LEMMA 18.1.6. *Let $w \in F = F(X)$. Then for any $g \in F/N$ and $x \in X$ the value of π_w on the edge $e = (g, gx)$ is equal to the coefficient in front of g in the Fox derivative $(\partial w / \partial x)^\mu \in \mathbb{Z}G$, i.e.,*

$$(\partial w / \partial x)^\mu = \sum_{g \in G, x \in X} \pi_w(g, gx)g$$

Proof. Follows by induction on the length of w from formulas (67). \blacksquare

Figure 18.1 is an example for $F = F(\{x_1, x_2\})$ and $N = F'$. Non-zero values of π_w are shown as weights on the edges (zero weights are omitted).

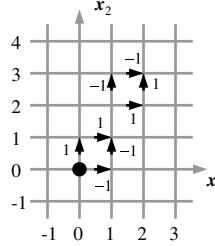


FIGURE 18.1. The values of π_w for $w = x_2 x_1 x_2 x_1 x_2 x_1^{-1} x_2^{-3} x_1^{-1}$ on (x_1, x_2) -grid. In this case $\partial w / \partial x_1 = -1 + x_2 - x_1 x_2^3 + x_1 x_2^2$ and $\partial w / \partial x_2 = 1 - x_1 + x_2^2 x_2^2 - x_1 x_2^2$.

The following theorem has been proven in [62, 278] using a homological argument similar to the one in Proposition 18.1.9. Here we give a short independent proof based on the Fox theorem.

THEOREM 18.1.7 ([62, 278]). *Let N be a normal subgroup of F and \sim_N an equivalence relation on F defined by*

$$u \sim_N v \iff \pi_u = \pi_v.$$

Then $F/N' = F/\sim_N$.

Proof. Let $u, v \in F$. Suppose $u = v$ in F/N' . Then $uv^{-1} \in N'$ hence by Fox theorem $\partial^\mu(uv^{-1})/\partial x = 0$ for every $x \in X$ (here by $\partial^\mu/\partial x$ we denote the canonical image of $\partial/\partial x$ in the group ring $\mathbb{Z}(F/N)$). It follows from (65) and (64) that

$$(69) \quad \frac{\partial}{\partial x}(uv^{-1}) = \frac{\partial u}{\partial x} - uv^{-1} \frac{\partial v}{\partial x}.$$

Hence in $\mathbb{Z}(F/N)$,

$$0 = \frac{\partial^\mu}{\partial x}(uv^{-1}) = \frac{\partial^\mu u}{\partial x} - \frac{\partial^\mu v}{\partial x},$$

so, by Lemma 18.1.6 $\pi_u = \pi_v$, as claimed.

To show the converse, notice first that $\pi_u = \pi_v$ implies that $u = v$ in F/N . Indeed, it can be seen from the definition of π but also follows from (63) and Lemma 18.1.6 since in this case

$$(u^\mu - v^\mu) - (u - v)^\varepsilon = \sum_{x \in X} \frac{\partial^\mu}{\partial x}(u - v) \cdot (x - 1) = 0$$

which implies $u^\mu = v^\mu$. Now by (69)

$$\frac{\partial^\mu}{\partial x}(uv^{-1}) = \frac{\partial^\mu u}{\partial x} - \frac{\partial^\mu v}{\partial x} = 0,$$

and, hence, by Fox theorem $uv^{-1} \in N'$. \blacksquare

REMARK 18.1.8. Theorem 18.1.7 relates the algebraic and geometric points of view on derivatives. One can prove this theorem (see Section 18.1.6) using a pure topological argument, then the Fox theorem, as well as, the Magnus embedding, come along as easy corollaries.

18.1.6. Geometric circulations and the first homology group of Γ . We describe here geometric circulations on $\Gamma = \Gamma(G, X)$ in a pure topological manner. For all required notions and results on homology of simplicial complexes we refer to [130] or [258].

In this section we view Γ as an infinite 1-complex.

PROPOSITION 18.1.9. *Let $G = F/N$, $\Gamma = \Gamma(G, X)$, and $\sigma : \pi_1(\Gamma) \rightarrow \mathcal{GC}(\Gamma)$ be a map defined by $\sigma(p) = \pi_p$ for $p \in \pi_1(\Gamma)$. Then:*

- (1) *σ is an epimorphism of groups;*
- (2) *every geometric circulation on Γ defines a 1-cycle on Γ ;*
- (3) *$H_1(\Gamma, \mathbb{Z}) \simeq \mathcal{GC}(\Gamma)$;*
- (4) *$\pi_1(\Gamma) \simeq N$ and $\ker \sigma = N'$.*

Proof. It was mentioned already in Section 18.1.4 that the map $p \rightarrow \pi_p$ is a morphism from the fundamental groupoid $\Pi(\Gamma)$ of paths in Γ into the groupoid of geometric flows $\mathcal{GF}(\Gamma)$. Hence, the restriction of this map onto the fundamental group $\pi_1(\Gamma)$ of Γ gives a homomorphism of groups. We have seen in Lemma 18.1.5 that σ is onto. This proves the first statement.

To see (2) observe first that a geometric circulation $f : E(\Gamma) \rightarrow \mathbb{Z}$, viewed as a formal sum $\sum_{e \in E(\Gamma)} f(e)e$, gives precisely a 1-chain in Γ (see, for example, [130]). Moreover, by definition, the net flow $f^*(v)$ at the vertex $v \in V(\Gamma)$ is the coefficient in front of v in the boundary ∂f of f . Therefore, $\partial f = 0$, so f is a 1-cycle.

(3) follows easily from the (2). Indeed, Γ is an 1-complex, so there are no non-trivial 1-boundaries in Γ . In this event $H_1(\Gamma, \mathbb{Z})$ is isomorphic to the group $\mathcal{GC}(\Gamma)$ of 1-cycles, as claimed.

(4) It is a classical result that the kernel of $\sigma : \pi_1(\Gamma) \rightarrow H_1(\Gamma, \mathbb{Z})$ is equal to the derived subgroup of $\pi_1(\Gamma)$ (see [130]). To prove (4) it suffices to notice that $\pi_1(\Gamma) \simeq N$, which is easy. ■

REMARK 18.1.10. Proposition 18.1.9 gives a simple geometric proof of Theorem 18.1.7, that relates the algebraic and geometric points of view on derivatives. Now one can derive the Fox theorem from the geometric argument above and then obtain the description of the kernel of the Magnus embedding, as an easy corollary.

18.1.7. Geodesics in F/N' . Let $G = F/N$ and $\mu : F(X) \rightarrow G$ the canonical epimorphism. In this section we describe geodesics of elements of the group $H = F/N'$ relative to the set of generators X^μ .

It is convenient to view the free group $F = F(X)$ as the set of all freely reduced words in the alphabet $X^{\pm 1} = X \cup X^{-1}$ with the obvious multiplication.

To describe geodesics in H (relative to X) of a given word $w \in F$ we need a construction from Lemma 18.1.5. Recall that p_w is the path in the Cayley graph $\Gamma = \Gamma(G, X)$ from 1 to w^μ with the label w and π_w the induced geometric flow on Γ with the source 1 and the sink w^μ . By Γ_w we denote the subgraph of Γ generated by $\text{supp}(\pi_w) \cup \{1, w^\mu\}$. Suppose Q is a finite subgraph of Γ such that $\Delta = \Gamma_w \cup Q$ is a connected graph and Q has the least number of edges among all such subgraphs. It follows from minimality of Q that every connected component of Q is a tree.

Moreover, if in the graph $\Delta = \Gamma_w \cup Q$ one collapses every connected component Γ_w to a point, then the resulting graph is a tree. We refer to Q as a *minimal forest* for w . In general, there could be several minimal forests for w .

Similarly, as in the proof of Lemma 18.1.5, we construct a finite X -digraph Δ^* by replicating every edge $e \in E(\Delta)$ with $|\pi_w(e)| - 1$ new edges in such a way that every vertex in $V(\Delta^*) - \{1, w^\mu\}$ has even directed degree and the map, that sends every replica of an edge e back to e (or e^{-1} depending on the orientation) is a morphism of X -labeled digraphs $\varphi : \Delta^* \rightarrow \hat{\Delta}$. There are two cases.

(CASE I) Suppose that p_w is a closed path in Γ , i.e., $w \in N$. In this case every vertex in Δ^* has even degree, so Δ^* has a Euler tour, say p_Q^* at 1. Denote by p_Q the image $\varphi(p_Q^*)$ of p_Q^* under φ . It follows from the construction (see Lemma 18.1.5) that p_Q is a closed path at 1 in Γ such that $\pi_w = \pi_{p_Q}$. Therefore, if $w_{p_Q} \in F$ is the label of p_Q then by Theorem 18.1.7 $w = w_{p_Q}$ in H . Moreover, since p_Q is an Euler tour in Δ^* its length, hence the length of w_{p_Q} is equal to

$$(70) \quad |p_Q| = \sum_{e \in \text{supp}(p_w)} |\pi_w(e)| + 2|E(Q)|.$$

(CASE II) Suppose that p_w is a not a closed path in Γ , i.e., $w \notin N$. By induction on $|w|$ it is easy to show that the vertices 1 and w^μ belong to the same connected component of Γ_w . Again, there exists an Euler's tour p_Q^* in the graph Δ^* which starts at the source and ends at the sink. Clearly, π_{p_Q} satisfies the equality (70). If u is the label of the path π_{p_Q} then $\pi_u = \pi_{p_Q} = \pi_w$ and u is a geodesic word for w .

Now, with the construction in place, we are ready to characterize geodesics in H of elements from N .

THEOREM 18.1.11. *Let $H = F/N'$ and $w \in F$. Then the following holds:*

- *If Q is a minimal forest for w then w_{p_Q} is a geodesic for w and*

$$l_X(w) = \sum_{e \in \text{supp}(p_w)} \pi_w(e) + 2|E(Q)|.$$

- *Every geodesic word for w is equal (in F) to a word w_{p_Q} for a suitable minimal forest Q and an Euler path p_Q^* .*

Proof. Let $u \in F$ be a geodesic word for w^μ in H . Observe, that $\Delta = \text{supp}(\pi_u) \cup p_u$ is a connected subgraph of Γ and

$$|p_u| \geq \sum_{e \in \text{supp}(p_u)} \pi_u(e) + 2|E(\Delta - \text{supp}(\pi_u))|.$$

Now, by Theorem 18.1.7 the equality $u = w$ in H implies $\pi_u = \pi_w$. In particular, $\text{supp}(\pi_u) = \text{supp}(\pi_w)$. Hence

$$(71) \quad |p_u| \geq \sum_{e \in \text{supp}(p_w)} \pi_w(e) + 2|E(\Delta - \text{supp}(\pi_w))|.$$

Since u is geodesic for w the number $|\Delta - \text{supp}(\pi_w)|$ is minimal possible, so $Q = \Delta - \text{supp}(\pi_w)$ is a minimal forest for w . In fact, the equation (71) shows that the converse is also true. This proves the theorem. ■

The discussion above shows that GP is easy in F/N' provided one can solve the following problem efficiently.

Minimal Forest Problem (MFP). Given a finite set of connected finite subgraphs $\Gamma_1, \dots, \Gamma_s$ in Γ find a finite subgraph Q of Γ such that $\Gamma_1 \cup \dots \cup \Gamma_s \cup Q$ is connected and Q has a minimal possible number of edges.

PROPOSITION 18.1.12. *GP in F/N' (relative to X) is linear time reducible to MFP for $\Gamma(F/N, X)$.*

Proof. Follows from the discussion above. Indeed, given a word $w \in F$ one can in linear time compute the flow π_w and find the connected components $\Gamma_1, \dots, \Gamma_s$ of $\text{supp}(\pi_w) \cup \{o(p_w), t(p_w)\}$ in $\Gamma = \Gamma(F/N, X)$. Then, solving MFP for these components, one gets the subgraph Q which makes the graph $\Gamma_1 \cup \dots \cup \Gamma_s \cup Q$ connected. Obviously, it takes linear time to find a Euler path in the graph Δ , hence to find a geodesic for w . ■

18.2. The word problem in free solvable groups

In this section we present fast algorithms to compute Fox derivatives of elements of a free group F in the group ring $\mathbb{Z}S_{r,d}$ of a free solvable group $S_{r,d}$. As an immediate application, we obtain a decision algorithm for WP in a free metabelian group M_r with time complexity $O(rn \log_2 n)$ and a decision algorithm for WP in $S_{r,d}$, $d \geq 3$, with time complexity $O(rdn^3)$. These are significant improvements in comparison with the known decision algorithms discussed in the Introduction and Section 18.1.2. As another application we get a fast algorithm to compute images of elements from $S_{r,d}$ under the Magnus embedding, which opens up an opportunity to use efficiently the classical techniques developed for wreath products.

18.2.1. The word problem in free metabelian groups. In this section we compute Fox derivatives of elements of F in the group ring $\mathbb{Z}(F/F')$. Then we apply this to WP in free metabelian groups.

Let $X = \{x_1, \dots, x_r\}$, $F = F_r = F(X)$, $M = M_r = F/F^{(2)}$, $A = A_r = F/F'$, and $\mu : F \rightarrow A$ the canonical epimorphism. All Fox derivatives in this section are computed in the ring $\mathbb{Z}A$.

Let $w \in F$. Then

$$\frac{\partial^\mu w}{\partial x_i} = \sum_{a \in A} m_{a,i} a, \quad m_{a,i} \in \mathbb{Z}.$$

One can encode all the derivatives $\partial^\mu w / \partial x_i$ in one mapping $M_w : A \times \{1, \dots, r\} \rightarrow \mathbb{Z}$, where $M_w(a, i) = m_{a,i}$. Let

$$\text{supp}(M_w) = \{(a, i) \mid M_w(a, i) \neq 0\},$$

and S_w the restriction of M_w onto $\text{supp}(M_w)$. To compute Fox derivatives of w we construct a sequence of finite maps $S_0 = \emptyset, S_1, \dots, S_n = S_w$, as we read w . On each step k we either extend the domain $\text{Dom}(S_k)$ of S_k or change the value of S_k on some element from $\text{Dom}(S_k)$. To do this we need a data structure which allows one to do the following operations efficiently:

- for a given (a, i) determine if $(a, i) \in \text{Dom}(S_k)$ or not;
- add (a, i) to $\text{Dom}(S_k)$ if $(a, i) \notin \text{Dom}(S_k)$ and define $S_k(a, i) = q$ for some $q \in \mathbb{Z}$;
- change the value of S_k on (a, i) if $(a, i) \in \text{Dom}(S_k)$.

Every element $a \in A$ can be written uniquely in the form $a = x_1^{\delta_1(a)} \dots x_r^{\delta_r(a)}$, where $\delta_i(a) \in \mathbb{Z}$, so one may use the tuple of coordinates $\delta(a) = (\delta_1(a), \dots, \delta_r(a))$ to represent a . It follows from the formula (67) for partial derivatives that for every $(a, i) \in \text{Dom}(S_w)$ the components $\delta_j(a)$ of $\delta(a)$ satisfy the inequality $|\delta_j(a)| \leq |w|$, as well as the values of S_w and, hence, $|S_w(a, i)| \leq |w|$. Therefore, it takes $\lceil \log_2(|w|+1) \rceil$ bits to encode one coordinate $\delta_j(a)$ (one extra bit to encode the sign + or -), $r \lceil \log_2(|w|+1) \rceil$ bits to encode $\delta(a)$, and at most $r \lceil \log_2(|w|+1) \rceil + \lceil \log_2 r \rceil$ bits to encode (a, i) . We denote the binary word encoding (a, i) by $(a, i)^*$.

Thus every function S_k can be uniquely represented by a directed $\{0, 1\}$ -labeled binary tree T_k with the root ε and with leaves labeled by integers such that $(a, i) \in \text{Dom}(S_k)$ if and only if there exists a path in T_k from the root ε to a leaf labeled by the code of (a, i) and such that the leaf of this path is labeled precisely by the integer $S_k(a, i)$. Notice, that the height of T_k is equal to $r \lceil \log_2(|w|+1) \rceil + \lceil \log_2 r \rceil$. Such a tree is visualized schematically in Figure 18.2.

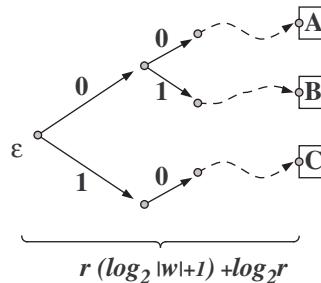


FIGURE 18.2. The tree T_k representing the function S_k .

REMARK 18.2.1. It is clear that one can perform every operation mentioned above on this data structure in at most $r \lceil \log_2(|w|+1) \rceil + \lceil \log_2 r \rceil$ elementary steps.

We use this data structure to design the following algorithm for computing S_w .

ALGORITHM 18.2.2 (Computing Fox derivatives in F/F').

INPUT. $r \in \mathbb{N}$ and $w = x_{i_1}^{\varepsilon_1} \dots x_{i_n}^{\varepsilon_n} \in F(X)$, where $i_j \in \{1, \dots, r\}$ and $\varepsilon_j = \pm 1$.

OUTPUT. S_w .

COMPUTATIONS.

- Set $S = \emptyset$ and $\delta(a) = (0, \dots, 0) \in \mathbb{Z}^r$.
- For $j = 1, \dots, n$ do:
 - if $\varepsilon_j = 1$ then
 - * check if there is a path (from the root to a leaf) labeled by $(\delta(a), i_j)^*$ in S ;
 - * if such a path does not exist in S then create it, add it to S , and put 1 as the corresponding value at the new leaf;
 - * if such a path exists in S then add 1 to the value at its leaf;
 - * add ε_j to the i_j th coordinate of $\delta(a)$;
 - if $\varepsilon_j = -1$ then
 - * add ε_j to the i_j th coordinate of $\delta(a)$;

- * check if there is a path (from the root to a leaf) labeled by $(\delta(a), i_j)^*$ in S ;
- * if such a path does not exist in S then create it, add it to S , and put -1 at the corresponding leaf;
- * if such a path exists in S then subtract 1 from the value at its leaf.

C. Output S_n .

THEOREM 18.2.3. *Given $r \in \mathbb{N}$ and $w \in F$, Algorithm 18.2.2 computes all partial derivatives of w in $\mathbb{Z}A$ (the mapping S_w) in time $O(r|w|\log_2|w|)$.*

Proof. Using the formula (67) for partial derivatives it is easy to check that given $w \in F$ Algorithm 18.2.2, indeed, computes the mapping S_w . To verify the complexity estimates observe first that Algorithm 18.2.2 performs exactly $|w|$ iterations at step [B.]. Each iteration requires $O(r[\log_2(|w|+1)])$ elementary steps (see Remark 18.2.1), so altogether one has $O(r|w|\log_2|w|)$ as the time complexity estimate for Algorithm 18.2.2, as claimed. ■

ALGORITHM 18.2.4 (Word problem in free metabelian groups).

INPUT. $r \in \mathbb{N}$ and $w \in F$.

OUTPUT. *True* if w represents the identity in M_r and *False* otherwise.

COMPUTATIONS.

- Apply Algorithm 18.2.2 to compute S_w .
- Check, looking at the values assigned to leaves of S_w , if all Fox derivatives $\partial w / \partial x_i$, $i = 1, \dots, r$ are equal to 0 or not.
- If all the derivatives are 0 then output *True*. If there is a non-zero derivative then output *False*.

THEOREM 18.2.5. *Algorithm 18.2.4 solves the word problem in a free metabelian group M_r in time $O(r|w|\log_2|w|)$.*

Proof. Follows from Theorem 18.2.3 and the Fox theorem (see Section 18.1.3). ■

18.2.2. The word problem in free solvable groups. In this section we present an algorithm to compute all Fox derivatives of a word $w \in F$ in the group ring $\mathbb{Z}S_{r,d-1}$, $d \geq 2$ in time $O(rd|w|^3)$. This gives a decision algorithm for WP in $S_{r,d}$ within time complexity $O(rdn|^3)$.

Let $X = \{x_1, \dots, x_r\}$, $F = F_r = F(X)$, $S = S_{r,d} = F/F^{(d)}$, $d \geq 3$, and $\mu : F \rightarrow S_{r,d-1}$ the canonical epimorphism. All Fox derivatives in this section are computed in the ring $\mathbb{Z}S_{r,d-1}$.

In Section 18.2.1 we used a unique representation of elements $a \in A_r$ by their coordinate vectors $\delta(a)$ to compute Fox derivatives in nearly linear time. Now we do not have such normal forms of elements of $S_{r,d}$, $d \geq 2$, so our computations are slightly different, however, the general strategy is quite similar. To speed up computations we use some data structures based on efficient partitioning techniques.

In general, let G be a group generated by X and D a finite subset of $F(X)$. A *G-partition* of D is a partition of D into a union of disjoint non-empty subsets D_i such that for any $u, v \in D$, $u = v$ in G if and only if they belong to some subset D_i . Clearly, the G -partition of D is unique. Observe that if a group H is a quotient of G then the G -partition of D is the same or *finer* than the H -partition of D .

If the set D is ordered, say $D = \{w_0, \dots, w_n\}$ then the G -partition of D can be represented by a function $P : \{0, \dots, n\} \rightarrow \{0, \dots, n\}$ where $P(j) = i$ if and only if $w_i = w_j$ in G and i is the smallest with such property. Given the G -partition P of D one can arrange this data in such a way that it takes linear time (in the size of j) to compute $P(j)$. In particular, given i, j one can check in linear time if $w_i = w_j$ in G . Also, for a given word $w \in D$ one can find in linear time an index i such that $w = w_i$. These are the main two subroutines concerning partitions of D .

Let $w = x_{i_1}^{\varepsilon_1} \dots x_{i_n}^{\varepsilon_n}$, where $i_j \in \{1, \dots, r\}$ and $\varepsilon_j = \pm 1$. Put

$$(72) \quad D_w = \{\varepsilon, x_{i_1}^{\varepsilon_1}, x_{i_1}^{\varepsilon_1} x_{i_2}^{\varepsilon_2}, \dots, x_{i_1}^{\varepsilon_1} \dots x_{i_n}^{\varepsilon_n}\} \subset F(X_r).$$

We order the set D_w as follows $w_0 = \varepsilon, \dots, w_n = w$. Now, to check whether or not the derivative $\partial w / \partial x_i$ is trivial in $\mathbb{Z}S_{r,d-1}$ one has to determine which pairs (w_i, w_j) of elements from D_w represent the same element in $S_{r,d-1}$ and then cancel out w_i with w_j in $\partial w / \partial x_i$ if they have the opposite signs.

The goal of Algorithm 18.2.10 below is to compute $S_{r,d}$ -partition for D_w . This is performed in a sequence of iterations. The algorithm starts out by computing the A_r -partition of D_w . On the second iteration the algorithm computes the M_r -partition of D_w . On the third step it computes the $S_{r,3}$ -partition of D_w . It continues this way until it computes the $S_{r,d}$ -partition of D_w .

To explain how the algorithm works assume that the $S_{r,d-1}$ -partition of D_w is given by the partition function P_{d-1} described above. Notice, that the $S_{r,d}$ -partition of D_w is the same or finer than the $S_{r,d-1}$ -partition of D_w , since $S_{r,d-1}$ is a quotient of $S_{r,d}$. This shows that to construct $S_{r,d}$ -partition P_d of D_w one has only to compare elements from D_w which are equal in $S_{r,d-1}$. Suppose that $w_s, w_t \in D_w$, $s < t$ and $w_s = w_t$ in $S_{r,d-1}$. To check if $w_s = w_t$ in $S_{r,d}$ we compare all their Fox derivatives in $\mathbb{Z}S_{r,d-1}$, so for every $k = 1, \dots, r$ we compute the following differences:

$$\begin{aligned} & \partial w_s / \partial x_k - \partial w_t / \partial x_k \\ = & \sum_{1 \leq j \leq s, i_j=k, \varepsilon_j=1} x_{i_1}^{\varepsilon_1} \dots x_{i_{j-1}}^{\varepsilon_{j-1}} - \sum_{1 \leq j \leq s, i_j=k, \varepsilon_j=-1} x_{i_1}^{\varepsilon_1} \dots x_{i_j}^{\varepsilon_j} \\ & - \sum_{1 \leq j \leq t, i_j=k, \varepsilon_j=1} x_{i_1}^{\varepsilon_1} \dots x_{i_{j-1}}^{\varepsilon_{j-1}} + \sum_{1 \leq j \leq t, i_j=k, \varepsilon_j=-1} x_{i_1}^{\varepsilon_1} \dots x_{i_j}^{\varepsilon_j} \\ = & - \sum_{s+1 \leq j \leq t, i_j=k, \varepsilon_j=1} x_{i_1}^{\varepsilon_1} \dots x_{i_{j-1}}^{\varepsilon_{j-1}} + \sum_{s+1 \leq j \leq t, i_j=k, \varepsilon_j=-1} x_{i_1}^{\varepsilon_1} \dots x_{i_j}^{\varepsilon_j} \\ (73) \quad = & - \sum_{s+1 \leq j \leq t, i_j=k, \varepsilon_j=1} w_{j-1} + \sum_{s+1 \leq j \leq t, i_j=k, \varepsilon_j=-1} w_j. \end{aligned}$$

Clearly, given w and s , as above one can compute the formal expression (73) in time $O(|w|)$. To check if (73), viewed as an element in $\mathbb{Z}S_{r,d-1}$, is equal to 0 it suffices to represent it in the standard group ring form $\sum_{g \in S_{r,d-1}} mg$ (where $m \in \mathbb{Z}$) and verify if all coefficients in this representation are zeros. Now we describe a particular procedure, termed Collecting Similar Terms Algorithm, which gives the standard group ring form for (73). Given (73) one can compute in time $O(|w|)$ the following sum:

$$(74) \quad - \sum_{s+1 \leq j \leq t, i_j=k, \varepsilon_j=1} w_{P(j-1)} + \sum_{s+1 \leq j \leq t, i_j=k, \varepsilon_j=-1} w_{P(j)}.$$

Observe now, that two summands w_p and w_q in (74) are equal in $S_{r,d-1}$ if and only if $p = q$. It is easy to see that it takes time $O(|w|)$ to collect similar terms in (74), i.e., to compute the coefficients in the standard group ring presentation of (74).

It follows that $\partial w_s / \partial x_k = \partial w_t / \partial x_k$ in $\mathbb{Z}S_{r,d-1}$ if and only if all the coefficients in the standard group ring form of (74) are equal to 0. The argument above shows that one can check whether or not $\partial w_s / \partial x_k = \partial w_t / \partial x_k$ in $\mathbb{Z}S_{r,d-1}$ in time $O(|w|)$. Since we need to compare all partial derivatives $\partial / \partial x_k$, $k = 1, \dots, r$, of the elements w_s and w_t it takes altogether $O(r|w|)$ time to verify if $w_s = w_t$ in $S_{r,d-1}$.

The routine above allows one to construct effectively the $S_{r,d}$ -partition P_d of D_w when given the $S_{r,d-1}$ -partition P_{d-1} . A more formal description of the algorithm is given below.

ALGORITHM 18.2.6 (Computing the $S_{r,d}$ -partition of D_w).

INPUT. Positive integers $r \geq 2$, $d \geq 2$, and a word $w \in F(X)$.

OUTPUT. The $S_{r,d}$ -partition function P_d for the set D_w .

INITIALIZATION. Compute the set D_w and form the initial (trivial) F -partition P_0 of D_w , so $P_0(i) = i$ for every $i \in \{0, \dots, n\}$.

COMPUTATIONS.

- A. Compute A -partition P_1 of D_w .
- B. For $c = 2, \dots, d$ do:
 - 1) For each $0 \leq s < t \leq n$ such that $w_s = w_t$ in $S_{r,c-1}$ check whether or not $w_s = w_t$ in $S_{r,c}$.
 - 2) Form the $S_{r,c}$ -partition P_d of D_w .
- C. Output P_d .

LEMMA 18.2.7. *Given integers $r, d \geq 2$ and $w \in F$ Algorithm 18.2.6 computes the $S_{r,d}$ -partition (the function P_d) of D_w in time $O(dr|w|^3)$.*

Proof. Algorithm 18.2.6 makes precisely d iterations $c = 1, \dots, d$ by consequently computing the $S_{r,c}$ -partitions of D_w . After the $S_{r,c-1}$ -partition of D_w is computed, the algorithm computes the $S_{r,c}$ -partition of D_w by comparing elements $w_s, w_t \in D_w$ in $S_{r,c}$. It requires at most $|w|(|w| + 1)/2$ such checks, and, as was explained above, every such check can be done in time $O(r|w|)$. Altogether one needs $O(r|w|^3)$ steps to construct the function P_c on the iteration c . Since the algorithm makes altogether d iterations it takes it $O(dr|w|^3)$ time to produce P_d , as claimed. ■

Now we are in a position to show two applications of Algorithm 18.2.6. The first is concerned with computing Fox derivatives in $\mathbb{Z}S_{r,d}$.

ALGORITHM 18.2.8 (Computing Fox derivatives).

INPUT. Positive integers $r \geq 2$, $d \geq 2$, a word $w \in F(X)$, and a number $k \in \{1, \dots, r\}$.

OUTPUT. The standard group ring presentation of the Fox derivative $\partial w / \partial x_k$ in $\mathbb{Z}S_{r,d}$.

COMPUTATIONS.

- A. Compute, using formals (67), the Fox derivative $\partial w / \partial x_k$ in $\mathbb{Z}F$.
- B. Compute, using Algorithm 18.2.6, the $S_{r,d}$ -partition of D_w .
- C. Compute, using the Collecting Similar Terms Algorithm, the standard group ring form of $\partial w / \partial x_k$ in $\mathbb{Z}S_{r,d}$.
- D. Output $\partial w / \partial x_k$ computed in [C].

LEMMA 18.2.9. *Given integers $r, d \geq 2$, a word $w \in F$, and a number $k \in \{1, \dots, r\}$ Algorithm 18.2.8 computes the standard group ring presentation of the Fox derivative $\partial w / \partial x_k$ in $\mathbb{Z}S_{r,d}$ in time $O(dr|w|^3)$.*

Proof. Follows from Lemma 18.2.7. ■

The second application of Algorithm 18.2.6 is to WP in $S_{r,d}$.

ALGORITHM 18.2.10 (Word problem in $S_{r,d}$).

INPUT. Positive integers $r \geq 2$, $d \geq 2$, and a word $w \in F(X)$.

OUTPUT. *True* if $w = 1$ in $S_{r,d}$ and *False* otherwise.

COMPUTATIONS.

- A. Compute the set D_w .
- B. Using Algorithm 18.2.6 compute $S_{r,d}$ -partition P_d of D_w .
- C. If $P_d(0) = P_d(n)$, i.e., $1 = w$ in $S_{r,d}$, then output *True*. Otherwise output *False*.

THEOREM 18.2.11. *Algorithm 18.2.10 solves the word problem in a free solvable group $S_{r,d}$ in time $O(dr|w|^3)$.*

Proof. Follows immediately from Lemma 18.2.7. ■

18.3. Geodesics in free metabelian groups

In this section we discuss the computational hardness of different variations of geodesic problems and prove the main result about **NP**-completeness of BGLP in free metabelian groups.

18.3.1. Algorithmic problems about geodesics in groups. Let G be a group with a finite set of generators $X = \{x_1, \dots, x_r\}$ and $\mu : F(X) \rightarrow G$ the canonical epimorphism. In this section we view the free group $F(X)$ as the set of all freely reduced words in the alphabet $X^{\pm 1} = X \cup X^{-1}$ with the usual multiplication.

For a word w in the alphabet $X^{\pm 1}$ we denote by $|w|$ the length of w . The *geodesic length* of an element $g \in G$ relative to X , denoted by $l_X(g)$, is the length of a shortest word $w \in F(X)$ representing g , i.e.,

$$l_X(g) = \min\{|w| \mid w \in F(X), w^\mu = g\}.$$

To simplify notation we write, sometimes, $l_X(w)$ instead of $l_X(w^\mu)$. A word $w \in F(X)$ is called *geodesic* in G relative to X , if $|w| = l_X(w)$. We are interested here in the following algorithmic problem for a given group G .

The Geodesic Problem (GP): Given a word $w \in F(X)$ find a geodesic (in G) word $\tilde{w} \in F(X)$ such that $w^\mu = \tilde{w}^\mu$.

Also, one can consider the following variation of GP.

The Geodesic Length Problem (GLP): Given a word $w \in F(X)$ find $l_X(w)$.

As customary in complexity theory one can modify GLP to get the corresponding decision problem:

The Bounded Geodesic Length Problem (BGLP): Let G be a group with a finite generating set X . Given a word $w \in F(X)$ and a natural number k determine if $l_X(w) \leq k$.

It was noticed in [67] that these three problems are polynomial-time reducible to each other. Also, it is easy to see that the word problem for G is polynomial-time reducible to GLP. The converse is not true (assuming $\mathbf{P} \neq \mathbf{NP}$), namely there exists a group with polynomial-time decidable word problem and \mathbf{NP} -complete geodesic search problem. The simplest example of this type is due to Perry, who showed in [220] that GLP is \mathbf{NP} -complete for a metabelian group $\mathbb{Z}_2 wr (\mathbb{Z} \times \mathbb{Z})$ (the wreath product of \mathbb{Z}_2 and $\mathbb{Z} \times \mathbb{Z}$). In the next section we clarify the situation with free metabelian groups.

It was claimed in [62] that in free solvable groups of finite rank GLP is decidable in polynomial time. Unfortunately, in this particular case their argument is fallacious. Our main result of this section is the following theorem.

THEOREM 18.3.1 (Main Theorem). *Let M_r be a free metabelian group M_r of finite rank $r \geq 2$. Then BGLP in M_r (relative to the standard basis) is \mathbf{NP} -complete.*

Proof. The proof of this result consists of two parts. First, in Section 18.3.2 (Corollary 18.3.5) we show that it suffices to prove that BGLP is \mathbf{NP} -complete in M_2 . Second, in Section 18.3.4 (Theorem 18.3.11) we give a proof that BGLP is, indeed, \mathbf{NP} -complete in M_2 . ■

This immediately implies the following results.

COROLLARY 18.3.2. *The search problems GP and GLP are \mathbf{NP} -complete in non-abelian M_r (relative to the standard basis).*

To prove the Main Theorem we reduce the problem to the case $r = 2$ and then show that BGLP in M_2 is \mathbf{NP} -complete. To see the latter we construct a polynomial reduction of the Rectilinear Steiner tree problem to BGLP in M_r .

18.3.2. Reduction to M_2 . Let \mathcal{V} be a variety of groups. For groups $A, B \in \mathcal{V}$ we denote by $A *_{\mathcal{V}} B$ the free product of A and B relative to \mathcal{V} . In particular, if $A = \langle X \mid R \rangle$, and $B = \langle Y \mid S \rangle$ are presentations of A and B in \mathcal{V} then $A *_{\mathcal{V}} B = \langle X \cup Y \mid R \cup S \rangle$ is a presentation of $A *_{\mathcal{V}} B$ in \mathcal{V} . As usual, $A *_{\mathcal{V}} B$ satisfies the canonical universal property: any two homomorphism $A \rightarrow C, B \rightarrow C$ into a group $C \in \mathcal{V}$ extends to a unique homomorphism $A *_{\mathcal{V}} B \rightarrow C$ (we refer to [213] for details). It follows that if $F_{\mathcal{V}}(X \cup Y)$ is a free group in \mathcal{V} with basis $X \cup Y$ then $F_{\mathcal{V}}(X \cup Y) = F_{\mathcal{V}}(X) *_{\mathcal{V}} F_{\mathcal{V}}(Y)$.

The following lemma claims that free \mathcal{V} -factors of a group G are isometrically embedded into G .

LEMMA 18.3.3. *Let $A, B \in \mathcal{V}$ with finite generating sets X and Y . Then in the group $A *_{\mathcal{V}} B$ no geodesic word (relative to $X \cup Y$) for an element from A contains a letter from Y . In particular, for any word $w \in F(X)$ its geodesic length in A (relative to X) is equal to the geodesic length in $A *_{\mathcal{V}} B$ (relative to $X \cup Y$).*

Proof. Let $w \in F(X)$ be a geodesic word in A relative to X . Suppose that $u \in F(X \cup Y)$ is a geodesic word in $G = A *_{\mathcal{V}} B$ (relative to $X \cup Y$) that defines the same element as w . The identical map $A \rightarrow A$ and the trivial map $B \rightarrow 1$ give rise to a homomorphism $\varphi : G \rightarrow A$. This φ , when applied to u , just “erases” all letters from Y . It follows that if u contains a letter from Y then $|u^{\varphi}| < |u| \leq |w|$ - contradiction with the assumption that w is geodesic in A relative to X (since $w = u^{\varphi}$ in A). ■

COROLLARY 18.3.4. *In the notation above, each of the problems GP, GLP, BGLP in A (relative to X) is polynomial time reducible to the problem of the same type in $A *_{\mathcal{V}} B$ (relative to $X \cup Y$).*

Notice, that $M_{n+m} = M_n *_{\mathcal{A}_2} M_m$, where \mathcal{A}_2 is the variety of all metabelian groups. Since WP is in P for groups from \mathcal{M} the corollary above implies the following result.

COROLLARY 18.3.5. *If BGLP is NP-complete in M_2 then it is NP-complete in $M_r, r \geq 2$, relative to the standard bases.*

REMARK 18.3.6. Corollary 18.3.5 easily generalizes to free groups in an arbitrary variety, provided they have WP decidable in polynomial time.

18.3.3. Rectilinear Steiner tree problem. The Steiner tree problem (STP), which was originally introduced by Gauss, is one of the initial twenty one NP-complete problems that appeared in the Karp's list [152]. We need the following *rectilinear* variation of STP.

Let \mathbb{R}^2 be the Euclidean plane and Γ the integer grid canonically embedded into \mathbb{R}^2 (all vertices from \mathbb{Z}^2 together with all the horizontal and vertical lines connecting them). If A is a finite subset of \mathbb{Z}^2 then a *rectilinear Steiner tree* (RST) for A is a subgraph T of Γ such that $A \cup T$ is connected in Γ . By $s(T)$ (size of T) we denote the number of edges in T . An RST for A is *optimal* if it has the smallest possible size among all RST for A , we denote such RST by T_A . Observe, that a given A may have several different optimal RST, but their size is the same, we denote it by $s(A)$.

Notice that, in general, T_A for A is not a spanning tree for A in Γ , it may contain some vertices from \mathbb{Z}^2 which are not in A (so-called, *Steiner points*). The Rectilinear Steiner tree problem (RSTP) asks for a given finite $A \subseteq \mathbb{Z}^2$ and $k \in \mathbb{N}$ decide if there exists some T_A for A with $s(T_A) < k$. It is known that RSTP is NP-complete [88].

18.3.4. NP-completeness of BGLP in M_2 . Now we construct a polynomial reduction of RSTP to GLDP in M_2 relative to the standard basis $X = \{x, y\}$.

With each point $(s, t) \in \mathbb{Z}^2$ we associate a word

$$w_{s,t} = x_1^s x_2^t \cdot (x_2 x_1 x_2^{-1} x_1^{-1}) \cdot x_2^{-t} x_1^{-s}$$

in $F(x, y)$. Similarly, with a set of points $A = \{(s_1, t_1), \dots, (s_n, t_n)\} \subset \mathbb{Z}^2$, ordered in an arbitrary way, we associate a word

$$w_A = \prod_{i=1}^n w_{s_i, t_i}.$$

Observe, that the word $w_{s,t}$, as well as w_A , belongs to $F' = [F, F]$, so in M_2 they define elements from M'_2 . In particular, the path p_{w_A} is a closed path in the grid $\Gamma = \mathbb{Z}^2$, which is viewed as the Cayley graph of the abelianization F/F' .

For $A \subset \mathbb{Z}^2$, $(p, q) \in \mathbb{Z}^2$, and $m \in \mathbb{Z}$ we put

- $A + (p, q) = \{(s + p, t + q) \mid (s, t) \in A\}$,
- $mA = \{(ms, mt) \mid (s, t) \in A\}$.

The following result is due to Hanan [128].

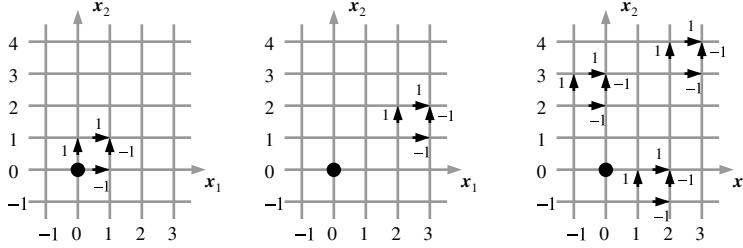


FIGURE 18.3. Flows on \mathbb{Z}^2 defined by words $w_{0,0}$, $w_{2,1}$, and $w_{\{(-1,2),(1,-1),(2,3)\}}$.

THEOREM 18.3.7 ([128], see Theorem 4). *Let $A = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be a finite subset of $\mathbb{Z} \times \mathbb{Z}$. There exists some T_A for A with the set of Steiner points $Q = \{(a_1, b_1), \dots, (a_q, b_q)\}$ such that $\{a_1, \dots, a_q\} \subseteq \{x_1, \dots, x_n\}$ and $\{b_1, \dots, b_q\} \subseteq \{y_1, \dots, y_n\}$.*

COROLLARY 18.3.8. *Let A be a finite subset of $\mathbb{Z} \times \mathbb{Z}$. Then*

- (1) $s(A + (b, c)) = s(A)$ for any $(b, c) \in \mathbb{Z}^2$;
- (2) $ms(A) = s(mA)$ for any $m \in \mathbb{N}$.

Proof. The first statement is obvious because the parallel shift $(x, y) \rightarrow (x, y) + (b, c)$ is an isomorphism of the Cayley graph Γ (in particular, an isometry).

To prove the second statement, notice first that $s(mA) \leq ms(A)$. Indeed, stretching T_A by the factor of m along all horizontal and vertical lines gives some RST for mA , hence the claim.

On the other hand, $s(A) \leq s(mA)/m$. To see this, observe that by Theorem 18.3.7 there exists $R = T_{mA}$ which lies inside the grid $m\mathbb{Z} \times m\mathbb{Z}$. Since the coordinates of all vertices in R are multiples of k one can shrink R by the factor of m , in such a way that the image of R becomes an RST for A . Clearly, the size of the image is equal to $s(T_{mA})/m$, hence the result. ■

PROPOSITION 18.3.9. *Let A be a finite subset of \mathbb{Z}^2 , $(b, c) \in A$, and $n = |A|$. Put $A^* = 10n(A - (b, c))$. Then $l_X(w_{A^*}) \in [20ns(A), 20ns(A) + 4n]$.*

Proof. Let u be a geodesic word for w_{A^*} relative to the basis X . Since $w_{A^*} \in F'$ the paths p_u and $p_{w_{A^*}}$ are closed paths in $\Gamma = \mathbb{Z}^2$ (viewed as the Cayley graph of M_2/M'_2). Hence u and w_{A^*} determine the same circulations $\pi_u = \pi_{A^*}$ on Γ . As described in Section 18.1.7, the flow π_u is associated with the subgraph Γ_u generated in Γ by $\text{supp}(\pi_u) \cup \{(0, 0)\}$. It follows from the construction of the word w_{A^*} that the connected components of Γ_u are precisely the 1×1 -squares in Γ , whose lower-left corners are located at the points from A^* . Notice that $(0, 0) \in A - (b, c)$ hence $(0, 0) \in A^*$. Now, if Q is a minimal forest for u (a subgraph of Γ of minimal size that makes the graph $\Gamma_u \cup Q$ connected in Γ) then by Theorem 18.1.11

$$(75) \quad |u| = l_X(w_{A^*}) = \sum_{e \in \text{supp}(p_u)} \pi_u(e) + 2|E(Q)| = 4n + 2|E(Q)|.$$

Observe, that an optimal RST T_{A^*} for A^* also makes the graph $\Gamma_u \cup Q$ connected in Γ , hence $|E(Q)| \leq s(A^*)$. Therefore, $l_X(w_{A^*}) \leq 20ns(A) + 4n$.

On the other hand assume that $|u| = l_X(w_{A^*}) < 20ns(A)$. Hence, from (75), there exists a minimal forest Q for A^* such that $2|E(Q)| < 20ns(A) - 4n$. Since every connected component has precisely 4 edges and there are n such components, it follows that there is an RST for A^* of size strictly less than $10ns(A)$ - contradiction with Corollary 18.3.8. This proves the proposition. ■

COROLLARY 18.3.10. *Let A be a finite subset of \mathbb{Z}^2 and $k \in \mathbb{N}$. In the notation above,*

$$s(A) < k \iff l_X(w_{A^*}) < 20nk + 4n.$$

In particular, this gives a polynomial reduction of RSTP to BGLP in M_2 relative to X .

Proof. Indeed, if $s(A) < k$ then by Proposition 18.3.9 $l_X(w_{A^*}) \leq 20ns(A) + 4n < 20nk + 4n$. On the other hand, suppose $s(A) \geq k$, say $s(A) = k+l$ for some positive $l \in \mathbb{N}$. Then, again by Proposition 18.3.9 $l_X(w_{A^*}) \geq 20ns(A) = 20n(k+l) > 20nk + 4n$, as required. ■

THEOREM 18.3.11. *GLDP in a free metabelian group M_2 is **NP**-complete.*

Proof. Corollary 18.3.10 gives a polynomial reduction of RSTP in \mathbb{Z}^2 to BGLP in M_2 . Therefore BGLP in M_2 is **NP**-hard. Meanwhile, as was mentioned above BGLP for M_2 is in **NP**, since the word problem for M_2 is polynomial-time decidable. ■

CHAPTER 19

Compressed Words

In order to prove upper bounds on the complexity of the word problem for some groups G , a “compressed” variant of the word problem for G was introduced in [173]. In the compressed word problem for G , the input word in the generators is not given explicitly but succinctly via a so called straight-line program (SLP for short). This is a context free grammar that generates exactly one word. Since the length of this word may grow exponentially with the size (number of productions) of the SLP, SLPs can be seen indeed as a succinct string representation. SLPs turned out to be a very flexible compressed representation of strings, which are well suited for studying algorithms for compressed data. In [174], [242] it was shown that the word problem for the automorphism group $\text{Aut}(G)$ can be reduced in polynomial time to the compressed word problem for G . In [242], it was shown that the compressed word problem for a finitely generated free group F_n can be solved in polynomial time. Hence, the word problem for $\text{Aut}(F_n)$ turned out to be solvable in polynomial time. This solved an open problem from [13], and this a motivating example for us in this chapter.

19.1. Straight line programs

DEFINITION 19.1.1. A *straight line program* (SLP) is a quadruple $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, where

- $X = \{x_1, \dots, x_n\}$ is a finite *alphabet*, or the set of *terminal symbols*;
- \mathcal{N} is a finite set of *non-terminal symbols*;
- $R \in \mathcal{N}$ is the *root symbol*;
- $\delta : \mathcal{N} \rightarrow X \cup (\mathcal{N} \times \mathcal{N})$ is the *set of production rules* such that for every $N \in \mathcal{N}$ either $\delta(N) = x \in X$, or $\delta(N) = AB \in \mathcal{N} \times \mathcal{N}$, defining an acyclic *production graph* $G(\mathcal{P})$ described below.

For an SLP \mathcal{P} we define a directed graph $G(\mathcal{P}) = (V, E)$ where $V = X \cup \mathcal{N}$ and $E = \{(N, x) \mid \delta(N) = x\} \cup \{(N, A) \mid \delta(N) = AB \text{ for some } B \in \mathcal{N}\} \cup \{(N, B) \mid \delta(N) = AB \text{ for some } A \in \mathcal{N}\}$. The graph $G(\mathcal{P})$ is *acyclic* if it does not contain a directed cycle.

For an SLP $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ inductively define a function $w : \mathcal{N} \rightarrow X^*$

$$w(N) = \begin{cases} \varepsilon & \text{if } \delta(N) = \varepsilon, \\ x & \text{if } \delta(N) = x \in X, \\ w(A)w(B) & \text{if } \delta(N) = AB. \end{cases}$$

and a function $\text{depth} : \mathcal{N} \rightarrow \mathbb{N}$

$$\text{depth}(N) = \begin{cases} 1 & \text{if } \delta(N) = x \in X \cup \{\varepsilon\}, \\ 1 + \max(\text{depth}(A), \text{depth}(B)) & \text{if } \delta(N) = AB. \end{cases}$$

Put

$$w(\mathcal{P}) = w(R) \text{ and } \text{depth}(\mathcal{P}) = \text{depth}(R).$$

The word $w(\mathcal{P})$ is called the *output* of \mathcal{P} . We want to point out that $w(\mathcal{P})$ belongs to a free monoid X^* even if X is a group alphabet, i.e., every symbol $x \in X$ has a formal inverse $x^{-1} \in X$. If X is a group alphabet and $w(N)$ is a reduced word for every $N \in \mathcal{N}$, then we say that \mathcal{P} is *reduced*.

LEMMA 19.1.2. *For every $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ and $N \in \mathcal{N}$, one has $|w(N)| \leq 2^{\text{depth}(N)}$. In particular, $|w(\mathcal{P})| \leq 2^{|\mathcal{P}|}$.* \square

Finally, for every $1 \leq \alpha \leq \beta \leq |w(\mathcal{P})|$ denote by $\mathcal{P}[\alpha : \beta]$ the restriction of $w(\mathcal{P})$ to the segment $[\alpha, \beta]$. Similarly, $\mathcal{P}[: \alpha] = \mathcal{P}[1 : \alpha]$ and $\mathcal{P}[\beta :] = \mathcal{P}[\beta : |w(\mathcal{P})|]$.

LEMMA 19.1.3 (On ε in productions). *The following statements hold:*

- If $\delta(N) = AB$ and $\delta(A) = \varepsilon$ then putting $\delta(N) = \delta(B)$ does not change the w function on \mathcal{N} and does not increase the values of depth function.
- If $\delta(N) = AB$ and $\delta(B) = \varepsilon$ then putting $\delta(N) = \delta(A)$ does not change the w function on \mathcal{N} and does not increase the values of depth function.
- If $\delta(N) = (A, p)$ and $\delta(A) = \varepsilon$ then putting $\delta(N) = \varepsilon$ does not change the w function on \mathcal{N} and does not increase the values of depth function.

\square

For $\mathcal{P}_1 = (X, \mathcal{N}_1, R_1, \delta_1)$ and $\mathcal{P}_2 = (X, \mathcal{N}_2, R_2, \delta_2)$ define $\mathcal{P}_1 \cup \mathcal{P}_2$ to be a GSLP $\mathcal{P} = (X, \mathcal{N}_1 \cup \mathcal{N}_2, R_1, \delta)$, where

$$\delta(N) = \begin{cases} \delta_1(N) & \text{if } N \in \mathcal{N}_1, \\ \delta_2(N) & \text{if } N \in \mathcal{N}_2. \end{cases}$$

Note that $\mathcal{P}_1 \cup \mathcal{P}_2 \neq \mathcal{P}_2 \cup \mathcal{P}_1$ in general.

19.2. Basic operations over SLPs

In this section we describe two basic operations over power circuits: taking a segment and inversion.

19.2.1. Subwords. Let \mathcal{P} be a straight line program. In this section we describe a series of algorithms. Every algorithm, given \mathcal{P} and some parameters, computes a new SLP \mathcal{P}' such that $w(\mathcal{P}')$ is a segment of $w(\mathcal{P})$.

ALGORITHM 19.2.1 ($(\mathcal{P}', R') = \text{InitialSegment}(\mathcal{P}, N, \alpha)$).

INPUT. A GSLP $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, $N \in \mathcal{N}$, and $\alpha \in \mathbb{Z}$ such that $1 \leq \alpha \leq |w(N)|$.

OUTPUT. A GSLP $\mathcal{P}' = (X, \mathcal{N}', R, \delta')$ such that $w(\mathcal{P}') = w(\mathcal{P})$, $\mathcal{N} \subseteq \mathcal{N}'$, and $\delta'|_{\mathcal{N}} = \delta$, and a new non-terminal R' satisfying $w(R') = N[: \alpha]$.

COMPUTATIONS.

- (1) Introduce a non-terminal R' and put $\delta(R') = \delta(N)$. Put $C = N$, $C' = R'$.
- (2) While $\delta(C) \notin X$
 - (a) Let $\delta(C) = AB$.
 - (b) If $\alpha < |w(A)|$, then put $\delta(C') = \delta(A)$ and $C = A$.
 - (c) If $\alpha = |w(A)|$, then put $\delta(C') = \delta(A)$.
 - (d) If $\alpha > |w(A)|$, then:
 - (i) Introduce a non-terminal N
 - (ii) Put $\delta(N) = \delta(B)$, $\delta(C') = AN$.

- (iii) Put $C = B$, $C' = N$, and $\alpha = \alpha - |w(N)|$.
(3) Return (\mathcal{P}, R') .

PROPOSITION 19.2.2. *Let \mathcal{P} be an SLP, $N \in \mathcal{N}$, and $\alpha \in \mathbb{Z}$ such that $1 \leq \alpha \leq |w(N)|$. If $(\mathcal{P}', R') = \text{InitialSegment}(\mathcal{P}, N, \alpha)$, then*

- (1) $w(R') = N[: \alpha]$,
- (2) $w(\mathcal{P}') = w(\mathcal{P})$,
- (3) $|\mathcal{P}'| \leq |\mathcal{P}| + \text{depth}(N)$,
- (4) $\text{depth}(\mathcal{P}') = \text{depth}(\mathcal{P})$.

Moreover, it requires $O(\text{depth}(N))$ arithmetic operations for Algorithm 19.2.1 to stop.

Proof. A single iteration of the loop (2) decreases the value of $\text{depth}(C)$ by at least 1 and, maybe, introduces one non-terminal N satisfying $\text{depth}(N) \leq \text{depth}(R)$. Hence, properties (2)–(4) and the complexity estimate hold. It is easy to see from the description of the algorithm that property (1) too. ■

ALGORITHM 19.2.3 ($(\mathcal{P}', R') = \text{TerminalSegment}(\mathcal{P}, N, \alpha)$).

INPUT. A GSLP $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, $N \in \mathcal{N}$, and $\alpha \in \mathbb{Z}$ such that $1 \leq \alpha \leq |w(N)|$.
OUTPUT. A GSLP $\mathcal{P}' = (X, \mathcal{N}', R, \delta')$ such that $w(\mathcal{P}') = w(\mathcal{P})$, $\mathcal{N} \subseteq \mathcal{N}'$, and $\delta'|_N = \delta$, and a new non-terminal R' satisfying $w(R') = N[\alpha :]$.

COMPUTATIONS.

- (1) Similar to Algorithm 19.2.1.

PROPOSITION 19.2.4. *Let \mathcal{P} be an SLP, $N \in \mathcal{N}$, and $\alpha \in \mathbb{Z}$ such that $1 \leq \alpha \leq |w(N)|$. If $(\mathcal{P}', R') = \text{TerminalSegment}(\mathcal{P}, N, \alpha)$, then*

- (1) $w(R') = N[\alpha :]$,
- (2) $w(\mathcal{P}') = w(\mathcal{P})$,
- (3) $|\mathcal{P}'| \leq |\mathcal{P}| + \text{depth}(N)$,
- (4) $\text{depth}(\mathcal{P}') = \text{depth}(\mathcal{P})$.

Moreover, it requires $O(\text{depth}(N))$ arithmetic operations for Algorithm 19.2.3 to stop. □

ALGORITHM 19.2.5 ($(\mathcal{P}', R') = \text{Segment}(\mathcal{P}, N, \alpha, \beta)$).

INPUT. A GSLP $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, $N \in \mathcal{N}$, and $\alpha, \beta \in \mathbb{Z}$ such that $0 \leq \alpha \leq \beta \leq |w(N)|$.

OUTPUT. A GSLP $\mathcal{P}' = (X, \mathcal{N}', R, \delta')$ such that $w(\mathcal{P}') = w(\mathcal{P})$, $\mathcal{N} \subseteq \mathcal{N}'$, and $\delta'|_N = \delta$, and a new non-terminal R' satisfying $w(R') = N[\alpha : \beta]$.

COMPUTATIONS.

- (1) $(\mathcal{P}'', R'') = \text{TerminalSegment}(\mathcal{P}, N, \alpha)$.
- (2) $(\mathcal{P}', R') = \text{InitialSegment}(\mathcal{P}'', R'', \beta - \alpha)$.
- (3) Return (\mathcal{P}', R') .

PROPOSITION 19.2.6. *Let \mathcal{P} be an SLP, $N \in \mathcal{N}$, and $\alpha \in \mathbb{Z}$ such that $1 \leq \alpha \leq |w(N)|$. If $(\mathcal{P}', R') = \text{Segment}(\mathcal{P}, N, \alpha, \beta)$, then*

- (1) $w(R') = N[\alpha : \beta]$,
- (2) $w(\mathcal{P}') = w(\mathcal{P})$,
- (3) $|\mathcal{P}'| \leq |\mathcal{P}| + 2 \text{depth}(N)$,
- (4) $\text{depth}(\mathcal{P}') = \text{depth}(\mathcal{P})$.

Moreover, it requires $O(\text{depth}(N))$ arithmetic operations for Algorithm 19.2.5 to stop. □

ALGORITHM 19.2.7 (Letter at position i : $x = \text{Letter}(\mathcal{P}, N, i)$).

INPUT. A GSLP $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, $N \in \mathcal{N}$, and $i \in \mathbb{Z}$ such that $1 \leq i \leq |w(N)|$.

OUTPUT. A letter $x \in X$ such that $x = N[i]$.

COMPUTATIONS.

- (1) Similar to Algorithm 19.2.1.

PROPOSITION 19.2.8. Let \mathcal{P} be an SLP, $N \in \mathcal{N}$, and $i \in \mathbb{Z}$ such that $1 \leq i \leq |w(N)|$. If $x = \text{Letter}(\mathcal{P}, N, i)$, then $x = N[i]$. Moreover, it requires $O(\text{depth}(N))$ arithmetic operations for Algorithm 19.2.7 to stop. \square

19.2.2. Inversion. Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ be an SLP. The next algorithm for every $N \in \mathcal{N}$ introduces into \mathcal{P} a new non-terminal denoted by N^{-1} satisfying $w(N^{-1}) = w(N)^{-1}$.

ALGORITHM 19.2.9 ($\mathcal{P}' = \text{PreparePowers}(\mathcal{P})$).

INPUT. A GSLP $\mathcal{P} = (X^{\pm 1}, \mathcal{N}, R, \delta)$.

OUTPUT. A GSLP $\mathcal{P}' = (X^{\pm 1}, \mathcal{N}', R', \delta')$ such that $w(\mathcal{P})^{-1} = w(\mathcal{P}')$.

COMPUTATIONS.

- (1) Define a set of non-terminals $\mathcal{N}^{-1} = \{A^{-1} \mid A \in \mathcal{N}\}$, here A^{-1} is a formal name for a new non-terminal corresponding to A .
- (2) Put $\mathcal{N}' = \mathcal{N} \cup \mathcal{N}^{-1}$.
- (3) For all $A \in \mathcal{N}$ do:
 - (a) If $\delta(A) = x \in X^{\pm 1}$, then put $\delta(A^{-1}) = x^{-1} \in X^{\pm 1}$.
 - (b) If $\delta(A) = BC$, then put $\delta(A^{-1}) = C^{-1}B^{-1}$.
- (4) Return \mathcal{P}' .

The following lemma follows immediately from the description of Algorithm 19.2.9.

LEMMA 19.2.10. Let \mathcal{P} ba an SLP and $\mathcal{P}' = \text{PreparePowers}(\mathcal{P})$. Then

- $w(\mathcal{P}') = w(\mathcal{P})$,
- for every $A \in \mathcal{N}'$ there exists $A' \in \mathcal{N}'$ such that $w(A) = w(A')^{-1}$,
- for every $A \in \mathcal{N}'$ if $\delta(A) = (A_1, s)$ then $s > 0$.

Moreover, it requires $O(|\mathcal{P}|)$ arithmetic operations for Algorithm 19.2.9 to stop. \square

19.3. Plandowski's algorithm

In this section we introduce a version of Plandowski's algorithm (see [226]) that decides whether the outputs of two SLPs are equal.

19.3.1. Assertions. In this section we define the main tool in Plandowski's algorithm: systems of assertions and methods of working with them.

DEFINITION 19.3.1. Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, $A, B \in \mathcal{N}$, and let $p \in \mathbb{Z}$ satisfy $0 \leq p < |w(A)|$. A triple

$$\mathcal{A} = (A, B, p)$$

is called an *assertion* for \mathcal{P} .

We distinguish 2 types of assertions: subword and overlap. We say that (A, B, p) is

- an *overlap assertion* if $|w(A)| \leq p + |w(B)|$, and
- a *subword assertion* otherwise;

see Figure 19.1. An overlap assertion (A, B, p) is *satisfied* if

$$A[p+1:] = B[:|w(A)|-p].$$

A subword assertion (A, B, p) is *satisfied* if

$$A[p+1:p+|w(B)|] = w(B).$$

It is convenient to depict an assertion as in Figure 19.1. Informally an assertion is satisfied if common parts of A and B in a picture coincide.

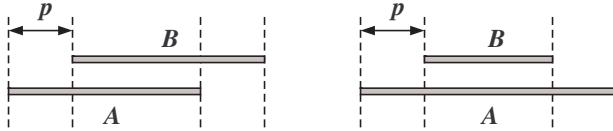


FIGURE 19.1. Overlap and subword and assertions.

We say that a set Γ of assertions for \mathcal{P} is satisfied if every assertion in Γ is satisfied. The empty set of assertions is satisfied by definition. Define

$$\text{Overlap}(\Gamma) = \{\mathcal{A} \in \Gamma \mid \mathcal{A} \text{ is an overlap assertion}\},$$

$$\text{Subword}(\Gamma) = \{\mathcal{A} \in \Gamma \mid \mathcal{A} \text{ is a subword assertion}\},$$

and

$$o(\Gamma) = |\text{Overlap}(\Gamma)| \text{ and } s(\Gamma) = |\text{Subword}(\gamma)|.$$

For an assertion $\mathcal{A} = (A, B, p)$ define

$$\text{depth}(\mathcal{A}) = \text{depth}(A) + \text{depth}(B).$$

LEMMA 19.3.2. *Let $\mathcal{P}_1 = (X, \mathcal{N}_1, R_1, \delta_1)$ and $\mathcal{P}_2 = (X, \mathcal{N}_2, R_2, \delta_2)$. Then $w(\mathcal{P}_1) = w(\mathcal{P}_2)$ if and only if the assertion $(R_1, R_2, 0)$ for $\mathcal{P}_1 \cup \mathcal{P}_2$ is satisfied.* \square

The idea of Plandowski's algorithm is to gradually reduce the assertion $(R_1, R_2, 0)$ to the empty set of assertions (or obtain an obviously unsatisfied assertion) using special transformations that we develop further in the sequel. Whenever we obtain a "simple" assertion in a current set of assertions we directly check whether or not it is satisfied. If it is not satisfied then the whole set is not satisfied. Otherwise we simplify the set by using the following rule.

LEMMA 19.3.3. *Let Γ be a set of assertions for \mathcal{P} and $\mathcal{A} \in \Gamma$. If \mathcal{A} is satisfied, then Γ is satisfied if and only if $\Gamma - \{\mathcal{A}\}$ is satisfied.* \square

An assertion $\mathcal{A} = (A, B, p)$ is called *simple* if $\delta(A) \in X$ or $\delta(B) \in X$.

LEMMA 19.3.4 (Satisfiability of simple assertions). *Let $\mathcal{A} = (A, B, p)$ be a simple assertion for \mathcal{P} . Then it takes $O(\text{depth}(B))$ arithmetic operations to check if \mathcal{A} is satisfied.*

Proof. Assume that $\delta(A) \in X$. Using Algorithm 19.2.7 we can compute $x = \text{Letter}(B, 1)$ and compare x with $\delta(A)$. It takes $O(\text{depth}(A))$ arithmetic operations to compute x . The case where $\delta(B) \in X$ is treated similarly. \blacksquare

Let Γ and Δ be sets of assertions for \mathcal{P} . We say that Γ and Δ are *equivalent* provided Γ is satisfied if and only if Δ is satisfied.

19.3.2. Trimming. In this section we define a special type of assertions which we call *trimmed* assertions. Intuitively, an assertion is trimmed if it does not trivially reduce to a “simpler” assertion. Formally, an assertion is not trimmed if it falls under one of the cases (T1)–(T6) described below; otherwise it is called trimmed. For every case we show how to simplify a non-trimmed assertion.

- (T1) If (A, B, p) is an overlap assertion, $\delta(A) = A_1 A_2$, and $p \geq |w(A_1)|$, then put $\mathcal{T}(\mathcal{A}) = (A_2, B, p - |w(A_1)|)$.
- (T2) If (A, B, p) is a subword assertion, $\delta(A) = A_1 A_2$, and $p + |w(B)| < |w(A_1)|$, then put $\mathcal{T}(\mathcal{A}) = (A_1, B, p)$.
- (T3) If (A, B, p) is a subword assertion, $\delta(A) = A_1 A_2$, and $p + |w(B)| = |w(A_1)|$, then put $\mathcal{T}(\mathcal{A}) = (A_1, B, p)$.
- (T4) If (A, B, p) is a subword assertion, $\delta(A) = A_1 A_2$, and $p = |w(A_1)|$, then put $\mathcal{T}(\mathcal{A}) = (B, A_2, 0)$.
- (T5) If (A, B, p) is a subword assertion, $\delta(A) = A_1 A_2$, and $|w(A_1)| < p$, then put $\mathcal{T}(\mathcal{A}) = (A_2, B, p - |w(A_1)|)$.
- (T6) If (A, B, p) is an overlap assertion, $\delta(B) = B_1 B_2$, and $|w(A)| \leq p + |w(B_1)|$, then put $\mathcal{T}(\mathcal{A}) = (A, B_1, p)$.

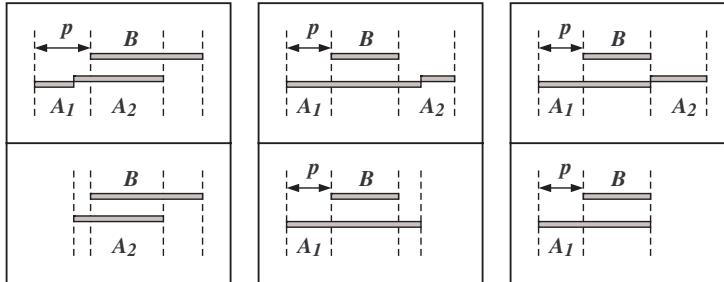


FIGURE 19.2. Trimming for cases (T1), (T2), and (T3).

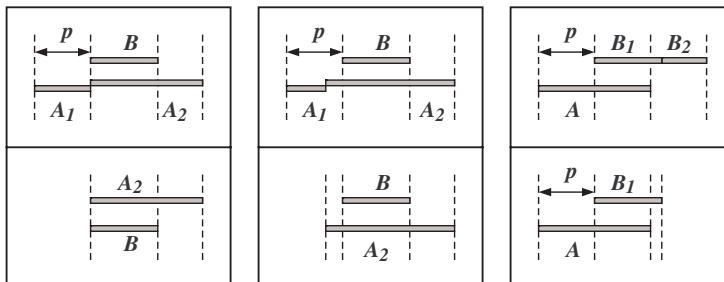


FIGURE 19.3. Trimming for cases (T4), (T5), and (T6).

PROPOSITION 19.3.5. Let $\mathcal{A} = (A, B, p)$ be a non-trimmed assertion and $\mathcal{A}' = \mathcal{T}(\mathcal{A})$. Then $\text{depth}(\mathcal{A}') < \text{depth}(\mathcal{A})$.

Proof. Every transformation (T1)–(T6) either replaces a base A with a base A' satisfying $\text{depth}(A') < \text{depth}(A)$, or replaces a base B with a base B' satisfying $\text{depth}(B') < \text{depth}(B)$. In either case $\text{depth}(\mathcal{A}') < \text{depth}(\mathcal{A})$. ■

If \mathcal{A} is a non-trimmed assertion then we apply an appropriate transformation described above and obtain a new assertion $\mathcal{A}_1 = \mathcal{T}(\mathcal{A})$. If \mathcal{A}_1 is not trimmed then we apply an appropriate transformation again and get $\mathcal{A}_2 = \mathcal{T}(\mathcal{A}_1)$, etc.

PROPOSITION 19.3.6. *If \mathcal{A} is a non-trimmed assertion then there exists a sequence of assertions $\mathcal{A} = \mathcal{A}_1, \dots, \mathcal{A}_n = \mathcal{T}^*(\mathcal{A})$, where $\mathcal{A}_{i+1} = \mathcal{T}(\mathcal{A}_i)$, such that*

- (1) $n \leq \text{depth}(\mathcal{A})$ and $\mathcal{T}^*(\mathcal{A})$ is trimmed;
- (2) for every $i = 1, \dots, n - 1$, \mathcal{A}_i is satisfied if and only if \mathcal{A}_{i+1} is satisfied;
- (3) if \mathcal{A} is an overlap assertion then $\mathcal{T}^*(\mathcal{A})$ is an overlap assertion.

Proof. (1) follows from Proposition 19.3.5. (2) and (3) follow from the definition of trimming. ■

For a set of assertions Γ define a set

$$\mathcal{T}^*(\Gamma) = \{\mathcal{T}^*(\mathcal{A}) \mid \mathcal{A} \in \Gamma\}.$$

COROLLARY 19.3.7. *For any set of assertions Γ ,*

- $\mathcal{T}^*(\Gamma)$ is equivalent to Γ ,
- $|\mathcal{T}^*(\Gamma)| \leq |\Gamma|$,
- $s(\mathcal{T}^*(\Gamma)) \leq s(\Gamma)$.

Moreover, computation of $\mathcal{T}^*(\Gamma)$ requires $O(|\Gamma| \cdot \max_{\mathcal{A} \in \Gamma} \text{depth}(\mathcal{A}))$ arithmetic operations. □

19.3.3. Splitting. In this section we introduce an operation that simplifies a single trimmed assertion \mathcal{A} . The operation is called *splitting*. For an assertion \mathcal{A} splitting produces an equivalent finite set of assertions $\mathcal{S}(\mathcal{A})$. There are two cases to consider (see Figure 19.4):

- If $\mathcal{A} = (A, B, p)$ is a trimmed overlap assertion and $\delta(B) = B_1 B_2$, then

$$\mathcal{S}(\mathcal{A}) = \begin{cases} \{(B_1, A, 0), (A, B_2, |w(B_1)|)\}, & \text{if } p = 0; \\ \{(A, B_1, p), (A, B_2, p + |w(B_1)|)\}, & \text{if } p > 0. \end{cases}$$

- If $\mathcal{A} = (A, B, p)$ is a trimmed subword assertion and $\delta(A) = A_1 A_2$ (note that $p > 0$ in this case), then

$$\mathcal{S}(\mathcal{A}) = \{(A_1, B, p), (B, A_2, |w(A_1)| - p)\}.$$

PROPOSITION 19.3.8. *Let $\mathcal{A} = (A, B, p)$ be a trimmed assertion. Then*

- (1) for every $\mathcal{A}' \in \mathcal{S}(\mathcal{A})$ we have $\text{depth}(\mathcal{A}') < \text{depth}(\mathcal{A})$,
- (2) \mathcal{A} is satisfied if and only if $\mathcal{S}(\mathcal{A})$ is satisfied,
- (3) $\mathcal{S}(\mathcal{A})$ contains up to one subword assertion,
- (4) $|\mathcal{S}(\mathcal{A})| \leq 2$.

Proof. Follows immediately from the definition of $\mathcal{S}(\mathcal{A})$. ■

For a set of assertions Γ we define the set of assertions

$$\mathcal{S}(\Gamma) = \bigcup_{\mathcal{A} \in \Gamma} \mathcal{S}(\mathcal{A}).$$

The following proposition immediately follows from the definition of splitting.

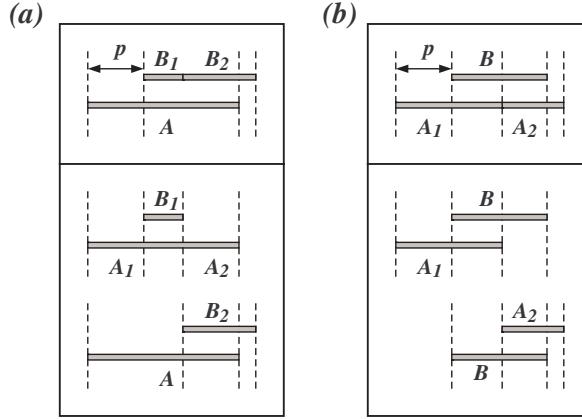


FIGURE 19.4. Splitting (a) trimmed overlap assertion (b) trimmed subword assertion.

PROPOSITION 19.3.9. *For any set of assertions Γ*

- $\mathcal{S}(\Gamma)$ is equivalent to Γ ,
- $|\mathcal{S}(\Gamma)| \leq 2|\Gamma|$,
- $s(\mathcal{S}(\Gamma)) \leq |\Gamma|$.

Moreover, it takes $O(|\Gamma|)$ arithmetic operations to compute $\mathcal{S}(\Gamma)$. \square

Below we describe a useful routine that converts a given trimmed subword assertion \mathcal{A} into an equivalent system $\{\mathcal{A}_1, \mathcal{A}_2\}$ of trimmed overlap assertions.

ALGORITHM 19.3.10 ($\Delta = \text{ConvertSubwordAssertion}(\mathcal{A})$).

INPUT. A trimmed subword assertion \mathcal{A} .

OUTPUT. A set of trimmed overlap assertions $\Delta = \{\mathcal{A}_1, \mathcal{A}_2\}$ equivalent to $\{\mathcal{A}\}$.
COMPUTATIONS.

- (1) Split \mathcal{A} and obtain a system $\{\mathcal{A}'_1, \mathcal{A}'_2\}$ of overlap assertions.
- (2) Put $\mathcal{A}_1 = \mathcal{T}^*(\mathcal{A}'_1)$ and $\mathcal{A}_2 = \mathcal{T}^*(\mathcal{A}'_2)$.
- (3) Return $\{\mathcal{A}_1, \mathcal{A}_2\}$.

Similarly, we can define an algorithm for converting a set of trimmed subword assertions Γ .

PROPOSITION 19.3.11. *Let \mathcal{A} be a trimmed subword assertion and*

$$\Delta = \text{ConvertSubwordAssertion}(\mathcal{A}).$$

Then $\Delta = \{\mathcal{A}_1, \mathcal{A}_2\}$ is an equivalent system of trimmed overlap assertions. Furthermore, Algorithm 19.3.10 terminates in $O(\text{depth}(\mathcal{A}))$ arithmetic operations. \square

19.3.4. Compactification. By Proposition 19.3.8, splitting simplifies a given system of assertions Γ . Therefore, if we continue applying splitting and trimming operations to a given system Γ , then eventually we will obtain a system of simple assertions that can be checked directly. Unfortunately, by Proposition 19.3.9 a single iteration of splitting can increase the size of Γ by a constant factor, which results in an exponential increase of $|\Gamma|$ with the number of iterations. To keep $|\Gamma|$

under control we introduce an operation called compactification. First, we prove a few auxiliary lemmas. All assertions in this section are trimmed overlap assertions.

DEFINITION 19.3.12. Let w be a word and $t \in \mathbb{N}$ satisfying $1 \leq t \leq |w|$. We say that w has *period* t if

$$w(i) = w(i + t)$$

for every $1 \leq i \leq |w| - t$.

LEMMA 19.3.13 (Periodicity Lemma, [176]). *If $t_1, t_2 \in \mathbb{N}$ are periods of w and $t_1 + t_2 \leq |w|$ then $\gcd(t_1, t_2)$ is also a period of w .* \square

Assertions (A, B, p_1) and (A, B, p_2) , with the same non-terminals A and B , are called *similar*.

LEMMA 19.3.14. *Let $\mathcal{A}_1 = (A, B, p_1)$, $\mathcal{A}_2 = (A, B, p_2)$, and $\mathcal{A}_3 = (A, B, p_3)$ be similar overlap assertions, satisfying*

$$(76) \quad \begin{cases} p_1 < p_2 < p_3, \\ p_2 - p_1 + p_3 - p_1 \leq |w(A)| - p_1. \end{cases}$$

Then the system of assertions $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ is equivalent to the system

$$\{\mathcal{A}_1, (A, B, p_1 + \gcd(p_2 - p_1, p_3 - p_1))\}.$$

Proof. The system $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ is satisfied if and only if \mathcal{A}_1 is satisfied and $A[:p_1+1]$ has periods $p_2 - p_1$ and $p_3 - p_1$. Since $p_2 - p_1 + p_3 - p_1 \leq |w(A)| - p_1$, the assumptions of Lemma 19.3.13 hold for $A[:p_1+1]$. Hence, $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ is satisfied if and only if \mathcal{A}_1 is satisfied and $A[:p_1+1]$ has period $\gcd(p_2 - p_1, p_3 - p_1)$ which is equivalent to the system $\{\mathcal{A}_1, (A, B, p_1 + \gcd(p_2 - p_1, p_3 - p_1))\}$. ■

DEFINITION 19.3.15. We say that a set of overlap assertions Γ for \mathcal{P} is *compact* if it does not contain 3 similar assertions $\mathcal{A}_1 = (A, B, p_1)$, $\mathcal{A}_2 = (A, B, p_2)$, and $\mathcal{A}_3 = (A, B, p_3)$, satisfying (76).

The idea of compactification is easy. If $|\Gamma|$ is relatively large, then Γ is not compact as we prove in Corollary 19.3.17. And the number of similar assertions in a non-compact system can be reduced as we described in Lemma 19.3.14.

LEMMA 19.3.16. *Let $\Gamma = \{\mathcal{A}_i\}_{i=1}^n$ be a system of similar overlap assertions $\mathcal{A}_i = (A, B, p_i)$. If $n > |\mathcal{P}|$ then Γ is not compact.*

Proof. We may assume that $p_i < p_{i+1}$ for every $i = 1, \dots, n-1$. It is easy to see that Γ is compact if and only if every subsystem $\{\mathcal{A}_i, \mathcal{A}_{i+1}, \mathcal{A}_{i+2}\}$ is compact. Define

$$\begin{aligned} \delta_i &= p_{i+1} - p_i, \text{ for } i = 1, \dots, n-1, \\ \delta_n &= |w(A)| - p_n. \end{aligned}$$

By definition $\{\mathcal{A}_i, \mathcal{A}_{i+1}, \mathcal{A}_{i+2}\}$ is compact if and only if $p_{i+2} - p_i + p_{i+1} - p_i \leq |w(A)| - p_i$ which is equivalent to

$$\delta_i \geq \delta_{i+1} + \dots + \delta_n + 1.$$

Clearly, $\delta_n \geq 1$. Using the inequality for δ_i we can deduce that if Γ is compact then $\delta_i \geq 2^{n-i}$. On the other hand, by Lemma 19.1.2, $|w(A)| \leq 2^{|\mathcal{P}|}$. Since $\delta_1 \leq |w(A)|$ it follows that if Γ is compact, then $n \leq |\mathcal{P}|$. ■

COROLLARY 19.3.17. *Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ and Γ is a set of trimmed overlap assertions for \mathcal{P} . If $|\Gamma| > 2|\mathcal{P}|^3$ then Γ is not compact.* \square

Let $\Gamma = \{\mathcal{A}_i\}_{i=1}^n$ be a system of similar overlap assertions $\mathcal{A}_i = (A, B, p_i)$. Lemma 19.3.14 suggests an algorithm on how to compute an equivalent compact set of assertions:

- Order p_i 's.
- If a triple $\mathcal{A}_i, \mathcal{A}_{i+1}, \mathcal{A}_{i+2}$ is not compact then replace it with a pair $\mathcal{A}_i, (A, B, p'_{i+1})$. It is important to note that the sequence of p_i 's remains ordered.

Clearly, this procedure can be performed in linear time $O(n)$. We consider gcd as an arithmetic operation.

ALGORITHM 19.3.18 ($\Delta = \text{Compactify}(\Gamma)$).

INPUT. A set Γ of trimmed overlap assertions.

OUTPUT. A compact set of assertions Δ equivalent to Γ .

COMPUTATIONS.

- (1) Order assertions.
- (2) Compactify every subset $\Gamma = \{\mathcal{A}_i\}_{i=1}^n$ of similar overlap assertions.
- (3) Call the result Δ and output it.

PROPOSITION 19.3.19. *Let Γ be a set of trimmed overlap assertions for \mathcal{P} . Algorithm 19.3.18 computes a compact (maybe non-overlap) set of assertions Δ in $O(|\Gamma| \log |\Gamma|)$ arithmetic operations.*

Proof. It requires $O(|\Gamma| \log |\Gamma|)$ operations to sort out elements in Γ and prepare subsets of similar assertions. Then it takes $O(|\Gamma|)$ operations to update non-compact triples. Correctness of Algorithm 19.3.18 follows from Lemma 19.3.14. ■

19.3.5. Satisfiability of SLP. Let \mathcal{A} be an assertion for \mathcal{P} . In this section we provide a polynomial time algorithm for checking if \mathcal{A} is satisfied or not.

ALGORITHM 19.3.20 ($S = \text{Satisfied}(\mathcal{P}, \mathcal{A})$).

INPUT. \mathcal{A} – assertion for $\mathcal{P} = (X, \mathcal{N}, R, \delta)$.

OUTPUT. Yes if Γ is satisfied, otherwise No.

COMPUTATIONS.

- (1) Put $\Gamma_0 = \{\mathcal{A}\}$ and $i = 0$.
- (2) While $\Gamma_i \neq \emptyset$ do:
 - (a) Compute $\Gamma' = \mathcal{T}^*(\mathcal{S}(\Gamma_i))$.
 - (b) Put $\Gamma'_o = \text{Overlap}(\Gamma')$ and $\Gamma'_s = \text{Subword}(\Gamma')$.
 - (c) Compute $\Gamma'' = \text{ConvertSubwordAssertion}(\Gamma'_s)$ using Algorithm 19.3.10.
 - (d) Put $\Delta = \mathcal{T}^*(\text{Compactify}(\Gamma'_o \cup \Gamma''))$.
 - (e) If some simple assertion $\mathcal{A} = (A, B, p)$ in Δ is not satisfied, then return No. Otherwise, remove all simple assertions from Δ .
 - (f) Put $\Gamma_{i+1} = \Delta$ and increase i by 1.
- (3) Return Yes.

THEOREM 19.3.21. *Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ be a reduced GSLP and \mathcal{A} a trimmed overlap assertion for \mathcal{P} . Algorithm 19.3.20 correctly decides if \mathcal{A} is satisfied, or not. Moreover, Algorithm 19.3.22 requires $O(|\mathcal{P}|^5)$ arithmetic operations to stop.*

Proof. In the loop (2) the algorithm constructs a sequence $\{\Gamma_i\}_{i=0}$ of sets of assertions. It follows from Lemma 19.3.3, Corollary 19.3.7, and Proposition 19.3.9,

that Γ_i is equivalent to Γ_{i+1} . Since we apply compactification at step (d), it follows from Corollary 19.3.17 that

$$|\Gamma_i| \leq 2|\mathcal{P}|^3.$$

Furthermore, since we apply splitting at step (a) it follows from Proposition 19.3.8 that

$$\max_{\mathcal{A} \in \Gamma_{i+1}} \{\text{depth}(\mathcal{A})\} < \max_{\mathcal{A} \in \Gamma_i} \{\text{depth}(\mathcal{A})\}.$$

Hence the algorithm performs at most $\max_{\mathcal{A} \in \Gamma} \{\text{depth}(\mathcal{A})\}$ iterations. Define

$$M := \max_{\mathcal{A} \in \Gamma} \{\text{depth}(\mathcal{A})\}.$$

Clearly, $M \leq 2|\mathcal{P}|$.

Now, when we have a bound on $|\Gamma_i|$ we can estimate complexity of a single iteration. Step (a) requires $O(|\mathcal{P}|^3)$ operations to split Γ_i (follows from Proposition 19.3.9) and $O(M|\mathcal{P}|^3)$ operations to trim the result (follows from Corollary 19.3.7). By Proposition 19.3.11, the conversion of a single subword assertion requires up to $O(M)$ operations. Hence, Step (c) requires $O(|\mathcal{P}|^4)$ operations to convert Γ'_s . Step (d) requires $O(|\mathcal{P}|^3 \log |\mathcal{P}|^3)$ operations by Proposition 19.3.19. Step (e) requires up to $O(M|\mathcal{P}|^2)$ operations by Lemma 19.3.4. Thus, the total complexity of 1 iteration can be bounded by $O(|\mathcal{P}|^4)$ arithmetic operations. Since the algorithm makes up to M iterations it follows that the total complexity is bounded by $O(|\mathcal{P}|^5)$ arithmetic operations. ■

19.3.6. Plandowski's algorithm. Here we combine all the preliminary results from the previous sections and describe Plandowski's algorithm.

ALGORITHM 19.3.22 ($E = \text{Equal}(A, B)$).

INPUT. $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ and $A, B \in \mathcal{N}$.

OUTPUT. E is Yes if $w(A) = w(B)$, otherwise No.

COMPUTATIONS.

- (1) If $|w(A)| \neq |w(B)|$, then return No.
- (2) Put $\Gamma_0 = \{(A, B, 0)\}$.
- (3) Return *Satisfied*(\mathcal{P}, Γ_0).

THEOREM 19.3.23 (Plandowski Algorithm). *Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ and $A, B \in \mathcal{N}$. Algorithm 19.3.22 correctly decides if $w(A) = w(B)$. Moreover, Algorithm 19.3.22 requires $O(|\mathcal{P}|^5)$ arithmetic operations to stop.*

Proof. Clearly, if $|w(A)| \neq |w(B)|$ then $w(A) \neq w(B)$. By Lemma 19.3.2 $w(A) = w(B)$ if and only if Γ_0 is satisfied. Now the result follows from Theorem 19.3.21. ■

19.4. Longest common initial segment

Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ be a reduced GSLP. For a triple (A, B, p) , where $A, B \in \mathcal{N}$, $p \in \mathbb{N}$ and $0 \leq p < |w(A)|$ define the value $CIS(A, B, p) \in \mathbb{N}$ to be the length of the longest common initial segment of $A[p+1:]$ and $w(B)$, i.e.,

$$CIS(A, B, p) = \min\{i \in \mathbb{N} \mid A[p+i+1] \neq B[i+1]\}.$$

Also, define $Overlap(A, B, p) \in \mathbb{N}$ to be

$$Overlap(A, B, p) = \min\{|w(A)| - p, |w(B)|\},$$

the length of the overlap for this triple. The value $CIS(A, B, p)$ can be computed recurrently as follows.

ALGORITHM 19.4.1 (Common initial segment: $L = CIS(A, B, p)$).

INPUT. A GSLP $\mathcal{P} = (X, \mathcal{N}, R, \delta)$, $A, B \in \mathcal{N}$, and $0 \leq p < |w(A)|$.

OUTPUT. The length L of the common initial segment of $A[p+1:]$ and $w(B)$.

COMPUTATIONS.

- (1) Let $\delta(A) = A_1A_2$.
- (2) If $|w(A_1)| \leq p$, then return $CIS(A_2, B, p - |w(A_1)|)$.
- (3) If $p + |w(B)| \leq |w(A_1)|$, then return $CIS(A_1, B, p)$.
- (4) If the assertion (A_1, B, p) is satisfied, then return $Overlap(A_1, B, p) + CIS(B, A_2, |w(A_1)| - p)$. Otherwise, return $CIS(A_1, B, p)$.

PROPOSITION 19.4.2. Let $\mathcal{P} = (X, \mathcal{N}, R, \delta)$ and $A, B \in \mathcal{N}$. Algorithm 19.4.1 computes the value $CIS(A, B, p)$ in $O(|\mathcal{P}|^6)$ ops.

Proof. On a single iteration Algorithm 19.4.1 recurrently reduces the value of $CIS(A, B, p)$ to $CIS(A', B', p')$, where

$$\text{depth}(A') + \text{depth}(B') = \text{depth}(A) + \text{depth}(B) - 1.$$

Therefore, it can perform at most $\text{depth}(A) + \text{depth}(B)$ reductions. On every reduction it checks if a single assertion is satisfied. Hence, by Theorem 19.3.23 each iteration requires $O(|\mathcal{P}|^5)$ ops. Hence, the claimed complexity bound. ■

19.5. Reduction

Let $X = \{x_1, \dots, x_n\}$ be a finite set. Put $X^{-1} = \{x_1^{-1}, \dots, x_n^{-1}\}$ and $X^{\pm 1} = X \cup X^{-1}$. Let \mathcal{P} be an SLP over the group alphabet $X^{\pm 1}$. The algorithm below computes an SLP \mathcal{P}' such that $w(\mathcal{P}')$ is reduced and $w(\mathcal{P}') = w(\mathcal{P})$ in $F(X)$.

ALGORITHM 19.5.1 (Reduction: $\mathcal{P}' = \text{Reduce}(\mathcal{P})$).

INPUT. A GSLP $\mathcal{P} = (X^{\pm 1}, \mathcal{N}, R, \delta)$.

OUTPUT. A reduced GSLP $\mathcal{P}' = (X^{\pm 1}, \mathcal{N}', R', \delta')$ equivalent to \mathcal{P} .

COMPUTATIONS.

- (1) Put $\mathcal{P}' = \text{PreparePowers}(\mathcal{P})$. Now $\mathcal{P}' = (X^{\pm 1}, \mathcal{N}', R, \delta)$, $\mathcal{N}' = \mathcal{N} \cup \mathcal{N}^{-1}$.
- (2) For all $A \in \mathcal{N}'$ from lower to higher depth do:
 - (a) Let $\delta(A) = A_1A_2$.
 - (b) Compute $l = CIS(A_1^{-1}, A_2, 0)$.
 - (c) Compute $A'_1 = \text{InitialSegment}(\mathcal{P}', A_1, |A_1| - l)$.
 - (d) Compute $A'_2 = \text{TerminalSegment}(\mathcal{P}', A_2, l)$.
 - (e) Put $\delta(A) = A'_1A'_2$.
- (3) Return \mathcal{P}' .

THEOREM 19.5.2 (Complexity of the reduction algorithm). For a given SLP \mathcal{P} , Algorithm 19.5.1 outputs a reduced SLP \mathcal{P}' satisfying the following conditions:

- (1) for every $A \in \mathcal{N}$ there exists $A' \in \mathcal{N}'$ such that $w(A) =_{F(X)} w(A')$,
- (2) for every $A' \in \mathcal{N}'$, $w(A')$ is a reduced parametric word over $X^{\pm 1}$.

Moreover, Algorithm 19.5.1 requires $O(|\mathcal{P}|^7)$ arithmetic operations to stop.

Proof. First, we apply *PreparePower*, namely introduce a new non-terminal A^{-1} for every non-terminal $A \in \mathcal{N}$ such that $w(A^{-1}) = w(A)^{-1}$. Let $A \in \mathcal{N}$. If $\delta(A) \in X$ then A is already reduced. If $\delta(A) = A_1A_2$ then to reduce A we need to

reduce A_1 and A_2 and then remove the cancellation in the middle. This is what we do in the loop (2). Since we consider non-terminals from lower to higher depths, it follows that when we consider A , A_1 and A_2 are already reduced. At step (2b) we compute the length of cancellation in A_1A_2 . At steps (2c) and (2d) we take initial segment of A_1 and the terminal segment of A_2 without the parts that cancel out in the product A_1A_2 . Finally, we redefine $\delta(A)$ to $A'_1A'_2$.

It is important to mention that at step (2b) we use a non-terminal A_1^{-1} . A_1^{-1} is a reduced non-terminal satisfying $w(A_1^{-1}) = w(A_1)^{-1}$. Clearly, such a non-terminal exists because we added all inverses at step (1). Later we create more non-terminals at steps (2c) and (2d) when we remove cancellation in some $\delta(A) = A_1A_2$, but the inverses of new non-terminals are also created when we remove cancellation in $\delta(A^{-1}) = A_2^{-1}A_1^{-1}$. Also, observe that we do not process new non-terminals, they are reduced as subwords of reduced non-terminals. Therefore, we need to process only $|\mathcal{P}|$ non-terminals.

Computation of *CIS* requires $O(|\mathcal{P}|^6)$ operations, computing segments takes linear time and introduces new non-terminals of lower depths. Therefore the total number of arithmetic operations required to stop is $O(|\mathcal{P}|^7)$. ■

19.6. The word problem in the automorphism group of a free group

Let $X = \{x_1, \dots, x_n\}$, $X^{-1} = \{x_1^{-1}, \dots, x_n^{-1}\}$, and $X^{\pm 1} = X \cup X^{-1}$. Let $F = F(X)$. Denote by *Aut*(F) the group of automorphisms of the free group F . It is well known that *Aut*(F_n) is generated by a finite set of “elementary” automorphisms, called the Nielsen automorphisms. There are three types of Nielsen automorphisms.

(N1) For every $1 \leq s \neq t \leq n$ an automorphism defined by

$$\begin{cases} x_i \mapsto x_i x_t & \text{if } i = s, \\ x_i \mapsto x_i & \text{if } i \neq s, \end{cases}$$

and an automorphism defined by

$$\begin{cases} x_i \mapsto x_t x_i & \text{if } i = s, \\ x_i \mapsto x_i & \text{if } i \neq s \end{cases}$$

are Nielsen automorphisms.

(N2) For every $1 \leq s \leq n$ an automorphism defined by

$$\begin{cases} x_i \mapsto x_i^{-1} & \text{if } i = s, \\ x_i \mapsto x_i & \text{if } i \neq s \end{cases}$$

is a Nielsen automorphism.

(N3) For every $1 \leq s \neq t \leq n$ an automorphism defined by

$$\begin{cases} x_i \mapsto x_t & \text{if } i = s, \\ x_i \mapsto x_i & \text{if } i = t, \\ x_i \mapsto x_i & \text{otherwise.} \end{cases}$$

is a Nielsen automorphism.

The inverses of the automorphisms above are essentially of the same type. We also call them Nielsen automorphisms.

Denote by **1** the trivial (identity) automorphism of F_n . The word problem for the group $\text{Aut}(F_n)$ is the following algorithmic problem: for a sequence of Nielsen automorphisms $\varphi_1, \dots, \varphi_k$ determine if a product

$$\varphi = \varphi_1^{\varepsilon_1}, \dots, \varphi_k^{\varepsilon_k}$$

defines the trivial automorphism **1**. Clearly, φ is trivial if and only if

$$\varphi(x) = x, \text{ for every } x \in X.$$

Unfortunately, it is computationally infeasible to compute $\varphi(x)$ directly since (in general) the length of $\varphi(x)$ is exponentially in k . Using straight-line programs helps here.

Let $\varphi_1, \dots, \varphi_k$ be a sequence of Nielsen automorphisms. Below we construct a sequence of pairs $\{(\mathcal{P}_i, f_i)\}_{i=0}^k$ where

- $\mathcal{P}_i = (X^{\pm 1}, \mathcal{N}_i, R_i, \delta_i)$ is an SLP;
- $f_i : \{1, \dots, n\} \rightarrow \mathbb{N}$

satisfying the following conditions:

$$(77) \quad w(N_{f_i(j)}) =_{F(X)} \varphi_i(\dots \varphi_1(x_j) \dots), \text{ for every } j = 1, \dots, n,$$

and for every $N \in \mathcal{N}_i$ there exists $N' \in \mathcal{N}$ such that

$$(78) \quad w(N)^{-1} =_{F(X)} w(N').$$

It is straightforward to construct (\mathcal{P}_0, f_0) , put

- $\mathcal{N}_0 = \{N_1, \dots, N_n\} \cup \{N'_1, \dots, N'_n\}$,
- $R_0 = N_1$,
- put $\delta_0(N_i) = x_i$ and $\delta_0(N'_i) = x_i^{-1}$, for every $i = 1, \dots, n$,
- f_0 is the identity map, i.e., $f_0(i) = i$.

Clearly, both conditions (77) and (78) are satisfied.

Assume that we have already constructed (\mathcal{P}_i, f_i) . The automorphism φ_{i+1} can be of 3 types. We consider only an automorphism of the type (N1), other cases can be done in a similar way. So, assume that φ_{i+1} is defined by

$$\begin{cases} x_i \mapsto x_i x_t & \text{if } i = s, \\ x_i \mapsto x_i & \text{if } i \neq s. \end{cases}$$

Introduce two new non-terminals N_z and N'_z (with an appropriate index z) into \mathcal{P}_i , put

$$\delta(N_z) = N_{f_i(s)} N_{f_i(t)} \text{ and } \delta(N'_z) = N'_{f_i(t)} N'_{f_i(s)},$$

and put $f_i(s) = z$. Denote the obtained SLP by \mathcal{P}_{i+1} and the obtained function by f_{i+1} . It is trivial to check that the obtained pair $(\mathcal{P}_{i+1}, f_{i+1})$ satisfies the conditions (77) and (78). Other automorphisms of type (N1) and their inverses can be considered similarly.

THEOREM 19.6.1. *There is an algorithm that solves the word problem in $\text{Aut}(F)$ in $O(k^7)$ arithmetic operations.*

Proof. Let $\varphi_1, \dots, \varphi_k$ be a sequence of Nielsen automorphisms. As described above it requires $O(k)$ operations to construct an SLP \mathcal{P}_k and a function $f_k : \{1, \dots, n\} \rightarrow \mathbb{N}$ satisfying conditions (77) and (78). It follows from the construction that $|\mathcal{P}| = 2(n+k)$. Applying Algorithm 19.5.1 we obtain an equivalent reduced SLP \mathcal{P}' such that for every $A \in \mathcal{N}$ there exists $A' \in \mathcal{N}'$ such that $w(A) =_{F(X)} w(A')$ and

$w(A')$ is reduced. By Theorem 19.5.2 it requires $O(|\mathcal{P}_k|^7)$ arithmetic operations to compute \mathcal{P}' . Clearly, it takes linear time $O(\mathcal{P})$ to check if $w(A') = x_i$ for every $A' \in \mathcal{N}'$. Hence, the total complexity is $O(n^7)$. ■

APPENDIX A

Probabilistic Group-based Cryptanalysis

NATALIA MOSINA

This appendix attempts to link probability theory and cryptanalysis by introducing new (probabilistic) machinery for security analysis of group-based protocols. Here we generalize classical probability laws (such as the strong law of large numbers, for instance) to combinatorial objects and show that these generalizations can lead to “unexpected probabilistic attacks” on existing cryptographic protocols. On a larger scale, our purpose is to extend a rigorous mathematical foundation for assessing security (or zero-knowledge property) of cryptosystems based on infinite groups. It would allow us to explore yet another dimension of cryptography — probabilistic group-based cryptanalysis.

The content of the following exposition is related, directly or indirectly, to Sections 1.6, 2.1, 2.3.2, 8.2, 4.1, and 5.1 in the present book.

A.1. Introduction

Random objects with values in groups and graphs are constantly dealt with in many areas of mathematics and theoretical computer science. In particular, such objects are very important in group-based cryptography (see [194] or [57] for introduction to the subject). Having the notion of the average for random group elements, generalized laws of large numbers for groups with respect to this average together with results on the rate of convergence in these laws would broaden the range of applications of random group objects from both theoretical and practical point of view. With a continuing development of group-based cryptography, availability of such tools for analysis of probability measures and their characteristics on groups becomes especially important.

The purpose of the present Appendix is, first, to develop these novel probabilistic tools for working with finitely generated groups in order to build a new mathematical framework for group-based cryptanalysis; second, to propose practical algorithms for computing expectations of random group elements in order to make the proposed theoretical tools practical, and, finally, to apply our new methodology to security analysis of a concrete authentication protocol, due to Sibert et al., in order to illustrate that our probabilistic machinery can be effective in real applications (see Section A.7 in this appendix). The novelty of the method proposed in Section A.7 is that it allows us to avoid attacking the protocol by trying to solve an underlying complex algebraic problem, namely, the conjugacy search problem; instead, we use a probabilistic approach, which, as shown below (see Sections A.7.7 and A.7.8), can be quite effective.

This work was partially supported by the PSC-CUNY Grant Award 60014-40 41.

To achieve these goals, the exposition below is divided into three parts. The first one (Sections A.2, A.3, and A.4) is purely theoretical; it deals with generalization of some fundamental results of classical probability theory to finitely generated groups. The second part (Sections A.5 and A.6) is computational in nature. The third and the last part (Section A.7) features the “probabilistic cryptanalysis” focusing on the particular example of Sibert et al. protocol ([256], 8.2, A.7.7).

The classical strong law of large numbers (SLLN) states that for independent and identically distributed (i.i.d.) real-valued random variables $\{\xi_i\}_{i=1}^\infty$,

$$(79) \quad \frac{1}{n} \sum_{i=1}^n \xi_i \rightarrow \mathbb{E}(\xi_1)$$

almost surely (with probability one) as $n \rightarrow \infty$, provided that the expectation $\mathbb{E}(\xi_1)$ is finite (see [19]). Depending on the assumptions one is willing to make there are different versions of the result, but, nevertheless, they all are known under the general name of *laws of large numbers*. It is intuitively clear that a sample (empirical) average should converge to a population (theoretical) average. On the other hand, the statement is indisputably very deep because it entails the possibility to acquire precise information about randomly-occurring phenomena, or, quoting A. V. Skorokhod [259], “it allows us to make reliable conclusions from random premises,” meaning that the average of a large number of random variables is “practically” non-random.

It is natural to ask a question about the existence of counterparts of this result for different topological spaces and/or algebraic structures, including groups. Starting from the middle of the last century, there has been ongoing research, following different trends, concerning the existence of such generalizations of the SLLN. One line of this research investigates random walks on groups (see Section A.2.1.5 for a brief list of relevant literature sources). The present exposition follows another direction of that research – the one which is concerned with the problem of averaging in arbitrary metric spaces (see Section A.2.1 for a brief historical background). We generalize classical probability results to a finitely generated group G by starting, in Section A.2.2, with a locally finite connected graph $\Gamma = (V(\Gamma), E(\Gamma))$ and introducing the concept of expectation (mean-set or average), denoted by $\mathbb{E}(\xi)$ or $\mathbb{E}(\mu)$, for random group elements $\xi : \Omega \rightarrow V(\Gamma)$ (where Ω is just the underlying probability space and μ is the induced measure on $V(\Gamma)$). We get the corresponding notion of a mean-set (average, expectation) for finitely generated groups via their Cayley graphs. Once we have the notion of mean-set for group-valued random elements, we notice that it satisfies the so-called “shift” property (see Proposition A.2.5); namely,

$$(80) \quad \mathbb{E}(g\xi) = g\mathbb{E}(\xi), \forall g \in G$$

which is analogous to the linearity property of a classical expectation for real-valued random variables. This property proves to be very useful in applications (see Section A.7.7 below). The reader can look ahead and consult the Section A.7.5 for a practical motivation of our theoretical developments. Next we prove the almost sure (with probability one) convergence, in some appropriate sense, of sample (empirical) means for group/graph random elements to the actual (theoretical) mean, thus, generalizing the classical law (79) and preserving its fundamental idea (see Section A.3). Several formulations of the strong law of large numbers on groups

are presented in Section A.3.2. To extend our probabilistic machinery for groups further, we supplement the SLLN with the concentration of measure inequalities, namely, with the analogues of Chebyshev and Chernoff-like bounds on the rate of convergence in the SLLN for random graph/group elements. This task is carried out in Section A.4.

In the second part of this Appendix (Section A.5), we explore different configurations of mean-sets in graphs and their applications to trees and free groups. We start with the observation that it is impossible for certain combinations of vertices to comprise mean-sets of some graphs. This leads to the notion of a so-called *cut-point* for a metric space, in general, and for a graph (Γ, d) , in particular. It turns out that the existence of a cut-point in Γ affects possible configurations of mean-sets, and this is a subject of the Cut-Point Lemma that we prove in Section A.5. The lemma is followed by a series of corollaries featuring applications to trees and free groups. More specifically, we prove that if Γ is a tree and μ is a probability measure on $V(\Gamma)$, then $|\mathbb{E}\mu| \leq 2$. From this, we deduce that if μ is a probability distribution on a free group F , then the number of elements in its mean-set cannot exceed two points, for any F -valued random element ξ . Moreover, the representation of a center-set for a free product of finitely generated groups becomes apparent. Section A.6 deals with computational problems and methods of computing sample mean-sets. In particular, we propose a Direct Descent algorithm for such computation and prove that this algorithm finds a central point for trees. In Section A.6.1 we perform series of simple experiments with free groups in which we compute the sample mean-sets of randomly generated samples of n random elements and observe the convergence of the sample mean-set to the actual mean.

In the final third part (Section A.7) of the exposition, we use the new ideas and machinery from A.2 and A.5 to explain how our probabilistic methodology works for group-based cryptanalysis in practice. We provide careful analysis of an existing group-based authentication scheme — Sibert et al. group-based (Feige-Fiat-Shamir type) authentication protocol — and show that the protocol is not computationally zero-knowledge, and, thus, is not secure. The analysis is supported by experiments, providing an illustration of how the private element s gets revealed. In addition, we provide experimental evidence that our approach is practical and can succeed even for groups with no efficiently computable length function such as braid groups. More specifically, after some preliminaries on zero-knowledge interactive proof systems and description of security assumptions of the protocol, we explain the idea of our new probabilistic *mean-set attack* and introduce the so-called *search problem* in infinite groups in Section A.7.5. In Section A.7.7 we formulate and prove mean-set attack principles under different assumptions. The mean-set attack principles give asymptotic bounds on the failure rate in the proposed mean-set attack. It is shown in Section A.7.7 that the probability of the failure can decrease linearly or exponentially, depending on the distribution μ . We also indicate in the same section that even if the proposed algorithm fails, we can still gain some information about the secret key of the Prover. In other words, the more rounds of the protocol are performed, the more information about the secret key we can gain. In Sections A.7.8.1 and A.7.8.2, we present results of our experiments with different key generations. At the end, in Section A.7.9, we discuss possible methods for defending against the mean-set attack. Among other things, this final section

intends to motivate further study of distributions on groups and computational properties of groups.

A.2. Probability theory on groups

In this part of the appendix, we discuss the notion of average (mean-set, expectation) for (graph-)group-valued random elements and consider different properties of this object. Next, we generalize the strong law of large numbers to finitely generated groups. Finally, we formulate and prove the analogues of the classical Chebyshev's inequality and Chernoff-like asymptotic bounds for random group elements.

The results presented below form a new mathematical framework for group-based cryptography and find their applications to security analysis of group-based authentication protocols (see Section A.7.7). The reader can look ahead and consult the Section A.7.5 for a practical motivation of our theoretical developments. Moreover, if the reader is very application-oriented and is mostly interested in how theories are applied, he or she can move on to the cryptographic part of this Appendix, Section A.7, and consult the theoretical results presented below as the need arises.

A.2.1. Probabilities for various spaces – overview. Below we give a brief account (not a full survey) of some developments concerning probabilities and mean-values for various spaces as well as some already existing generalizations of the strong law of large numbers in order to highlight several stages of research that preceded the present work. The reader willing to proceed to the core theoretical results right away may skip this historical overview and move on to Section A.2.2.

A.2.1.1. Linear spaces. In 1935, Kolmogorov [162] proposed to study probabilities in Banach spaces. Later, the interpretation of stochastic processes as random elements in certain function spaces inspired the study of laws of large numbers for random variables taking values in linear topological spaces. Banach spaces fit naturally into the context of the strong law of large numbers because the average of n elements x_1, \dots, x_n in a Banach space is defined as $n^{-1}(x_1 + \dots + x_n)$. In addition, Banach space provides convergence, and Gelfand–Pettis integration provides the notion of expectation of a random element (see [94] and [224]). It goes as follows. Let X be a linear space with norm $\|\cdot\| : X \rightarrow \mathbb{R}$ and X^* is the topological dual of X . A random X -element ξ is said to have the expected value $\mathbb{E}(\xi) \in X$ if

$$\mathbb{E}(f(\xi)) = f(\mathbb{E}(\xi))$$

for every $f \in X^*$. Let $\{\xi_i\}_{i=1}^\infty$ be a sequence of random X -elements. Without loss of generality, we may assume that $\mathbb{E}\xi_i = 0$ for every i . The strong law of large numbers in a separable Banach space X for a sequence of i.i.d. random X -elements $\{\xi_i\}_{i=1}^\infty$ is first proved in [206]. It states that

$$\lim_{n \rightarrow \infty} \|n^{-1}(\xi_1 + \dots + \xi_n)\| = \mathbb{E}(\xi_1) = 0$$

with probability one. The strong law of large numbers for i.i.d. random elements in a Fréchet space was proved in [2]. A few other works discussing generalizations of the strong law of large numbers in linear spaces are [14], [15], [269].

A.2.1.2. *Metric spaces.* Unlike in linear spaces, in a general (non-linear) topological space X , one has to find some other ways to introduce the concept of averaging and expectation. In 1948, Fréchet, [82], proposed to study probability theory in general metric spaces and introduced a notion of a mean (sometimes called *Fréchet mean*) of a probability measure μ on a complete metric space (X, d) as the minimizer of $Ed^2(x, y)$, where

$$Ed^2(x, y) = \int_X d^2(x, y) \mu(dy)$$

when it exists and is unique. If the minimizer is not unique, then the set of minimizers can be considered. If $\xi : \Omega \rightarrow X$ on a given probability space $(\Omega, \mathcal{F}, \mathbf{P})$ (see [19], [75]) is a random element in (X, d) and if for some $\mathbf{x} \in X$,

$$(81) \quad Ed^2(\xi, \mathbf{x}) = \inf_{y \in X} Ed^2(\xi, y) < \infty,$$

then \mathbf{x} is called an *expected element* of X . These generalizations were not met with much enthusiasm at the time (see historical remarks on probabilities in infinite dimensional vector spaces in [107]), and Fréchet's suggestions, due to the lack of their applications, underwent rather slow developments in the middle of the last century.

Let us briefly mention some existing works on generalizing the classical SLLN to a metric space X . Let $\{\xi_i\}_{i=1}^\infty$ be a sequence of i.i.d. random elements with values in X . Let the expectation be defined as in (81), written as a set

$$\mathbb{E}(X) = \left\{ x \in X \mid Ed^2(\xi, x) = \inf_{y \in X} Ed^2(\xi, y) \right\}.$$

Define an empirical mean (average) of elements $\xi_1(\omega), \dots, \xi_n(\omega)$ to be the set

$$\mathbf{M}(\xi_1, \dots, \xi_n) = \left\{ x \in X \mid \sum_{i=1}^n d^2(x, \xi_i(\omega)) = \inf_{y \in X} \sum_{i=1}^n d^2(y, \xi_i(\omega)) \right\}.$$

One of the first works on generalization of the SLLN for metric spaces is given in 1977 by Ziezold ([286]). Ziezold considers a separable quasi-metric space X with a finite quasi-metric d . For a sequence of i.i.d. random X -elements $\{\xi_i\}_{i=1}^\infty$ such that $Ed^2(\xi, x)$ is finite for at least one $x \in X$, he proves that inclusion

$$(82) \quad \mathbf{M}(\omega) = \overline{\bigcap_{k=1}^{\infty} \bigcup_{n=k}^{\infty} \mathbf{M}(\xi_1(\omega), \dots, \xi_n(\omega))} \subseteq \mathbb{E}(\xi_1)$$

holds with probability one. Here, $\overline{\bigcup_{n=k}^{\infty} \mathbf{M}(\xi_1(\omega), \dots, \xi_n(\omega))}$ is the closure of the union of $\mathbf{M}(\xi_1(\omega), \dots, \xi_n(\omega))$'s. He also shows that, in general, the equality does not hold (for a finite quasi-metric space). In 1981, Sverdrup-Thygeson ([267]) proves inclusion (82) for compact connected metric spaces and shows that the equality does not hold in general (for a metric space) when the minimizer in (81) is not unique. In 2003, Bhattacharya and Patrangenaru in [17, Theorem 2.3] prove equality in (82) for the unique minimizer in (81) for metric spaces X such that every closed bounded subset of X is compact, improving Ziezold's and Sverdrup-Thygeson's results.

Manifolds. As the need for statistical analysis for spaces with differential geometric structure was arising, statistical inference on Riemannian manifolds started to develop rapidly, especially due to applications in statistical theory of shapes and

image analysis. These applications evolve around the concept of averaging. See [154] for an introduction into shape theory.

There are two main approaches to averaging of elements on a manifold. Every Riemannian manifold X is a metric space and hence one can use constructions from the previous section to define the notion of a mean. Fréchet mean of a probability measure on a manifold is also known as an *intrinsic mean* [17]. Non-uniqueness of the intrinsic mean is a source of different technical problems. Also, the intrinsic mean, even when unique, is often very difficult to compute in practice.

On the other hand, a manifold X can also be looked at as a submanifold of some Euclidean space \mathbb{R}^k and one can define a mean relative to this inclusion. If $\tau : X \rightarrow \mathbb{R}^k$ is an embedding of X into Euclidean space (\mathbb{R}^k, d_0) , then the extrinsic mean of a probability measure μ on a manifold X with respect to τ is the Fréchet mean associated with the restriction to $\tau(X)$ of the Euclidean distance on \mathbb{R}^k . In [17] the authors prove the strong law of large numbers for the intrinsic and extrinsic means on manifolds.

A.2.1.3. K -means. A notion of a mean (or a mean-set) can be generalized into k -mean. Let B be a Banach space with a norm $\|\cdot\|$ and $k \in \mathbb{N}$. For a set $H = \{h_1, \dots, h_k\}$ we define a partition of B as follows

$$S_i = \{x \in B \mid \|x - h_i\| \leq \|x - h_j\| \text{ for every } j = 1, \dots, k\} \setminus (S_1 \cup \dots \cup S_{i-1})$$

where $i = 1, \dots, k$ and a function $\pi_H : B \rightarrow B$,

$$\pi_H(x) = \sum_{i=1}^k h_i \cdot \mathbf{1}_{S_i}(x)$$

where $\mathbf{1}_{S_i}$ is the indicator function of S_i . Fix a suitable non-decreasing function $\Phi : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ (e.g., $\Phi(x) = x$ or $\Phi(x) = x^2$) and for a probability measure μ on B define a number

$$M(H) = \int_{x \in B} \Phi(\|x - \pi_H(x)\|) d\mu(x).$$

A set H_0 of k elements that minimizes the value of M is called a k -mean of a probability distribution μ . In general, there can be several minimizers, which leads to technical complications. In 1988, Cuesta and Matran ([51]) proved that empirical k -means converge to the k -mean H_0 of μ under the assumption that the k -mean is unique.

It is straightforward to generalize a notion of a k -mean to a general metric space (X, d) . Indeed, if we put

$$S_i = \{x \in B \mid d(x, h_i) \leq d(x, h_j) \text{ for every } j = 1, \dots, k\} \setminus (S_1 \cup \dots \cup S_{i-1})$$

and

$$M(H) = \int_{x \in B} \Phi(d(x, \pi_H(x))) d\mu(x),$$

then we get a similar notion. This type of k -means, with $\Phi(x) = x$, was considered by Rubinshtein in 1995 in [239] where it was called the k -center.

As we can see, in general, depending on the research goals, one can define mean values on a given metric space (X, d) using any powers of d , i.e., instead of dealing with minimization of $Ed^2(\xi, x)$ in (81), one can work with a very similar functional by minimizing $Ed^r(\xi, x)$ for any $r > 0$ if necessary.

A.2.1.4. Probabilities on algebraic structures. Metrics and probabilities on algebraic structures have been studied from different perspectives. One source to look at is the book of M. Gromov [120]. The reader can find some applications of Fréchet mean in statistical analysis of partially ranked data (such as elements of symmetric groups and homogeneous spaces) in the book of Diaconis ([60]). An extensive historical background of the studies of probabilities on algebraic structures is given in [107], where the author considers probabilities for stochastic semigroups, compact and commutative stochastic groups, stochastic Lie groups, and locally compact stochastic groups employing the techniques of Fourier analysis to obtain limit theorems for convolutions of probability distributions. The reader interested in the question of defining probabilities on groups can find several approaches to this issue in [28].

A.2.1.5. Random walks on groups. One way to generalize the strong law of large numbers for groups is to study the asymptotic behavior of the products $g_1 g_2 \dots g_n$, where $\{g_i\}_{i=1}^{\infty}$ is a sequence of i.i.d. random group elements, the so-called random walk on a group. The reader can consult [285] for an introduction to random walks on groups. In 1960, Furstenberg and Kesten ([84]) prove the generalization of the strong law of large numbers for random matrices. Namely, they show that the limit

$$\lim_{n \rightarrow \infty} \frac{\log \|g_1 g_2 \dots g_n\|}{n}$$

exists with probability one, with some restrictive conditions on the entries of g_i , without computing the limit explicitly. In 1963, Furstenberg solved this problem for normalized products of random matrices in terms of stationary measures ([83]). Computational techniques that would allow us to compute these measures are investigated in [225]. A concise account of a number of illuminating results in the direction of the generalization of the SLLN to groups can be found in [151], where the authors prove the theorem about the directional distribution of the product $g_1 g_2 \dots g_n$, thus, proving a general law of large numbers for random walks on general groups. The authors call it a multiplicative ergodic theorem or a general, noncommutative law of large numbers (see [151] for the precise statement).

A.2.2. Mean (expectation) of a group-valued random element. Let $(\Omega, \mathcal{F}, \mathbf{P})$ be a given probability space. In this section, we define the notion of expectation for graph random elements in the sense of Fréchet mean set, which is one of the possible ways to look at mean values (see Section A.2.1.2). The same definition will hold for random elements on groups. We also discuss properties of the mean sets on groups. In particular, we prove that for our expectation \mathbb{E} , we have $\mathbb{E}(g\xi) = g\mathbb{E}(\xi)$, $g \in G$.

A.2.2.1. The mean set in a graph. Let $\Gamma = (V(\Gamma), E(\Gamma))$ be a locally finite connected graph. A random Γ -element ξ is a measurable function $\xi : \Omega \rightarrow V(\Gamma)$. The random element ξ induces an atomic probability measure $\mu : V(\Gamma) \rightarrow [0, 1]$ on $V(\Gamma)$ in a usual way:

$$(83) \quad \mu(v) = \mu_{\xi}(v) = \mathbf{P}(\{\omega \in \Omega \mid \xi(\omega) = v\}), \quad v \in V(\Gamma).$$

Next, we introduce a *weight function* $M_{\xi} : V(\Gamma) \rightarrow \mathbb{R}$ by

$$M_{\xi}(v) = \mathbb{E}d^2(v, \xi) = \sum_{s \in V(\Gamma)} d^2(v, s)\mu(s),$$

where $d(v, s)$ is the distance between v and s in Γ . If $M_\xi(v)$ is finite, then we say that the *weight function* M_ξ is defined at v . The domain of M is the set

$$\text{domain}(M) = \left\{ v \in V(\Gamma) \mid \sum_{s \in V(\Gamma)} d^2(v, s) \mu_\xi(s) < \infty \right\}.$$

The case of interest of course, is when $M_\xi(v)$ is *totally defined*, meaning that $M_\xi(v)$ is finite at every $v \in V(\Gamma)$.

DEFINITION A.2.1. Let ξ be a random Γ -element such that M is totally defined. The set of vertices $v \in \Gamma$ that minimize the value of M ,

$$(84) \quad \mathbb{E}(\xi) = \{v \in V(\Gamma) \mid M(v) \leq M(u), \forall u \in V(\Gamma)\},$$

is called the *mean-set* (or the *center-set*, or *average*) of ξ .

Very often we leave the random element ξ in the background to shorten the notation and write $M(v)$ instead of $M_\xi(v)$. Moreover, we write $\mathbb{E}(\mu)$ instead of $\mathbb{E}(\xi)$ sometimes and speak of the mean set of distribution μ induced by ξ on $V(\Gamma)$.

LEMMA A.2.2. Let Γ be a connected graph, ξ a random Γ -element, and u, v adjacent vertices in Γ . If $M(u) < \infty$, then $M(v) < \infty$.

Proof. Easily follows from the definition of M and the triangle inequality. Indeed, Since $d(u, v) = 1$, by the triangle inequality for $v, u, i \in V(\Gamma)$ we have $d(v, i) \leq d(u, i) + d(u, v) = d(u, i) + 1$ and

$$\begin{aligned} M(v) &= \sum_{i \in V(\Gamma)} d^2(v, i) \mu(i) \leq \sum_{i \in V(\Gamma)} [d(u, i) + 1]^2 \mu(i) \leq \sum_{i \in V(\Gamma), d(u, i)=0} \mu(i) \\ &+ \sum_{i \in V(\Gamma), d(u, i) \geq 1} [d(u, i) + 1]^2 \mu(i) \leq 1 + 4 \sum_{i \in V(\Gamma), d(u, i) \geq 1} d^2(u, i) \mu(i) = 1 + 4M(u). \end{aligned}$$

■

COROLLARY A.2.3. Let Γ be a connected graph and ξ a random Γ -element. Then either $\text{domain}(M) = V(\Gamma)$ or $\text{domain}(M) = \emptyset$.

LEMMA A.2.4. Let Γ be a connected locally finite graph and ξ a random Γ -element. If M_ξ is totally defined, then $0 < |\mathbb{E}(\xi)| < \infty$.

Proof. Let μ be a probability measure on Γ induced by ξ . For an arbitrary but fixed vertex $v \in \Gamma$, the weight function

$$M(v) = \sum_{i \in V(\Gamma)} d^2(v, i) \mu(i) = \sum_{n=0}^{\infty} \left(n^2 \sum_{i \in V(\Gamma), d(v, i)=n} \mu(i) \right)$$

is defined at v by assumption. Choose $r \in \mathbb{N}$ such that

$$\frac{1}{2}M(v) \leq \sum_{n=0}^r \left(n^2 \sum_{i \in V(\Gamma), d(v, i)=n} \mu(i) \right) = \sum_{i \in B_v(r)} d^2(v, i) \mu(i),$$

where

$$(85) \quad B_v(r) = \{i \in V(\Gamma) \mid d(v, i) \leq r\}$$

is the *ball* in Γ of radius r centered at v . If we take a vertex u such that $d(u, v) \geq 3r$, then using the triangle inequality, we obtain the following lower bound:

$$\begin{aligned} M(u) &= \sum_{i \in V(\Gamma)} d^2(u, i)\mu(i) \geq \sum_{i \in B_v(r)} [2r]^2\mu(i) + \sum_{i \notin B_v(r)} d^2(u, i)\mu(i) \\ &\geq 4 \sum_{i \in B_v(r)} d^2(v, i)\mu(i) \geq 2M(v). \end{aligned}$$

Thus, $d(v, u) \geq 3r$ implies $u \notin \mathbb{E}(\xi)$ and, hence, $\mathbb{E}(\xi) \subseteq B_v(3r)$. Since the graph Γ is locally finite, it follows that the sets $B_v(3r)$ and $\mathbb{E}(\xi)$ are finite. This implies that the function M attains its minimal value in $B_v(3r)$ and hence $\mathbb{E}(\xi) \neq \emptyset$. ■

A.2.3. The mean set in a group. Let G be a group and $X \subseteq G$ a finite generating set for G . The choice of X naturally determines a distance d_X on G via its Cayley graph $C_G(X)$. Hence, Definition A.2.1 gives us a notion of a mean set for a random G -element. It follows from the definition of the distance d_X that for any $a, b, g \in G$ the equality

$$(86) \quad d_X(a, b) = d_X(ga, gb)$$

holds. This equality implies that $\mathbb{E}(\xi)$ possesses the desirable property $\mathbb{E}(g\xi) = g\mathbb{E}(\xi)$, as the following proposition shows.

PROPOSITION A.2.5 (“Shift” Property). *Let G be a group and $g \in G$. Suppose that $(\Omega, \mathcal{F}, \mathbf{P})$ is a given probability space and $\xi : \Omega \rightarrow G$ a G -valued random element on Ω . Then for the random element ξ_g defined by $\xi_g(\omega) = g\xi(\omega)$ we have $\mathbb{E}(\xi_g) = g\mathbb{E}(\xi)$.*

Proof. Let μ_{ξ_g} be the measure induced on G by ξ_g , in the manner of (83). It follows from the definition of ξ_g that for any $h \in G$

$$\begin{aligned} \mu_{\xi_g}(h) &= \mathbf{P}(\{\omega \mid \xi_g(\omega) = h\}) = \mathbf{P}(\{\omega \mid g\xi(\omega) = h\}) \\ &= \mathbf{P}(\{\omega \mid \xi(\omega) = g^{-1}h\}) = \mu_\xi(g^{-1}h). \end{aligned}$$

This, together with (86), implies that for any $h \in G$

$$\begin{aligned} M_{\xi_g}(h) &= \sum_{i \in G} d^2(h, i)\mu_{\xi_g}(i) = \sum_{i \in G} d^2(g^{-1}h, g^{-1}i)\mu_\xi(g^{-1}i) \\ &= \sum_{i \in G} d^2(g^{-1}h, i)\mu_\xi(i) = M_\xi(g^{-1}h). \end{aligned}$$

Hence, the equality $M_{\xi_g}(h) = M_\xi(g^{-1}h)$ holds for any random element ξ and $g, h \in G$. Therefore, for any $h_1, h_2 \in G$, $M_{\xi_g}(h_1) < M_{\xi_g}(h_2) \Leftrightarrow M_\xi(g^{-1}h_1) < M_\xi(g^{-1}h_2)$ and

$$\begin{aligned} \mathbb{E}(\xi_g) &= \left\{ h \in G \mid M_{\xi_g}(h) \leq M_{\xi_g}(f), \forall f \in G \right\} \\ &= \left\{ h \in G \mid M_\xi(g^{-1}h) \leq M_\xi(g^{-1}f), \forall f \in G \right\} \\ &= \left\{ h \in G \mid M_\xi(g^{-1}h) \leq M_\xi(f), \forall f \in G \right\} \\ &= \left\{ gh \in G \mid M_\xi(h) \leq M_\xi(f), \forall f \in G \right\} = g\mathbb{E}(\xi). \end{aligned}$$

■

The equality $d_X(a, b) = d_X(ag, bg)$ does not hold for a general group $G = \langle X \rangle$; it holds for abelian groups.

A.2.4. Other possible definitions of \mathbb{E} . There are other possible definitions of \mathbb{E} for which the statement of Proposition A.2.5 (and other results of Section A.3) holds. Let c be a positive integer. By analogy to the function $M_\xi(v)$, define a *weight function* $M_\xi^{(c)}(v)$ of class c by

$$M_\xi^{(c)}(v) = \sum_{i \in V(\Gamma)} d^c(v, i)\mu(i)$$

and the mean-set $\mathbb{E}^{(c)}(\xi)$ of class c to be

$$\mathbb{E}^{(c)}(\xi) = \{v \in V(\Gamma) \mid M^{(c)}(v) \leq M^{(c)}(u), \forall u \in V(\Gamma)\}.$$

It is straightforward to check that all the statements of the previous section hold for $M_\xi^{(c)}$ and $\mathbb{E}^{(c)}(\xi)$. In fact, it is not hard to see that when $c = 1$, we have a counterpart of the median of the distribution μ . It is easy to check that $\mathbb{E} = \mathbb{E}^{(2)}$ agrees with the classical definition of the expectation on \mathbb{Z} .

REMARK A.2.6. Observe that $\mathbb{E}^{(2)}$ does not coincide with the classical mean in \mathbb{R}^2 . Recall that the classical mean in \mathbb{R}^2 is defined coordinatewise, i.e., the mean of $(x_1, y_1), \dots, (x_n, y_n)$ is a point in \mathbb{R}^2 defined by $(\mathbb{E}X, \mathbb{E}Y)$. For example, consider the distribution on \mathbb{Z}^2 such that $\mu(0, 0) = \mu(0, 3) = \mu(3, 0) = 1/3$ and for all other points $\mu = 0$. Then the classical mean is the point $(1, 1)$, and the mean-set $\mathbb{E}^{(2)}$ is the point $(0, 0)$. See Figure A.1.

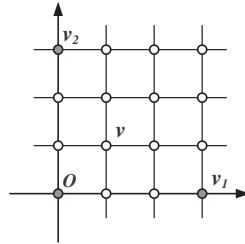


FIGURE A.1. $\mathbb{E}^{(2)}$ does not coincide with the classical mean in \mathbb{R}^2 .

A.3. Strong law of large numbers

Let ξ_1, \dots, ξ_n be a sample of independent and identically distributed (i.i.d.) graph-valued random elements $\xi_i : \Omega \rightarrow V(\Gamma)$ defined on a given probability space $(\Omega, \mathcal{F}, \mathbf{P})$. For every $\omega \in \Omega$, let $\mu_n(u; \omega)$ be the *relative frequency*

$$(87) \quad \mu_n(u; \omega) = \frac{|\{i \mid \xi_i(\omega) = u, 1 \leq i \leq n\}|}{n}$$

with which the value $u \in V(\Gamma)$ occurs in the random sample $\xi_1(\omega), \dots, \xi_n(\omega)$. We shall suppress the argument $\omega \in \Omega$ to ease notation, and let

$$M_n(v) = \sum_{i \in V(\Gamma)} d^2(v, i)\mu_n(i)$$

be the *sampling weight*, corresponding to $v \in V(\Gamma)$, and M_n the resulting *sampling weight function*.

DEFINITION A.3.1. The set of vertices

$$\mathbb{S}_n = \mathbb{S}(\xi_1, \dots, \xi_n) = \{v \in V(\Gamma) \mid M_n(v) \leq M_n(u), \forall u \in V(\Gamma)\}$$

is called the *sample mean-set* (or *sample center-set*, or *average*) of the vertices ξ_1, \dots, ξ_n .

Our goal is to prove that our (empirical) sample mean-set \mathbb{S}_n converges, in some sense, to the (theoretical) mean-set $\mathbb{E}(\xi)$ as $n \rightarrow \infty$. Before we go on, let us quickly recall the notions of limit-superior *limsup* and the limit-inferior *liminf* for a given sequence of sets.

Let $\{V_n\}_{n=1}^{\infty}$ be a sequence of subsets of V . Recall that

$$\limsup_{n \rightarrow \infty} V_n = \{v \mid v \in V_{n_k}, k = 1, 2, \dots\}$$

for some subsequence $\{n_k\}$ depending on v . We write that $\limsup_{n \rightarrow \infty} V_n = \{v \mid v \in V_n, \text{ i.o.}\}$, where i.o. stands for “infinitely often.” Similarly,

$$\begin{aligned} \liminf_{n \rightarrow \infty} V_n &= \{v \mid v \in V_n \text{ for all } n \text{ except for a finite number}\} \\ &= \{v \mid v \in V_n \text{ for all } n \geq n_0(v)\}. \end{aligned}$$

Properties of limits of sets can be found in numerous sources in the literature, [19] in particular. The simple example below helps to gain an intuition about the case when mean-sets contain more than one vertex.

EXAMPLE A.3.2. Consider a graph $\Gamma = (V, E)$ where $V = \{v_1, v_2\}$ and $E = \{(v_1, v_2)\}$, i.e., the graph Γ is the connected graph on 2 vertices. Let μ be the probability measure on $V(\Gamma)$ induced by some random element $\xi_1 : \Omega \rightarrow V$ and defined by $\mu(v_1) = \mu(v_2) = 1/2$. In that case $M(v_1) = M(v_2) = 1/2$ and, consequently, $\mathbb{E}(\xi) = \{v_1, v_2\}$.

Consider a sequence ξ_1, ξ_2, \dots of such random elements (independent and identically distributed). In other words, we just deal with a sequence of randomly generated vertices v_1, v_2 of the graph Γ . By definition,

$$M_n(v_1) = \frac{1}{n} |\{i \mid \xi_i = v_2, 1 \leq i \leq n\}| \quad \text{and} \quad M_n(v_2) = \frac{1}{n} |\{i \mid \xi_i = v_1, 1 \leq i \leq n\}|.$$

Hence,

$$v_1 \in \mathbb{S}_n \Leftrightarrow M_n(v_1) \leq M_n(v_2) \Leftrightarrow |\{i \mid \xi_i = v_2, 1 \leq i \leq n\}| \leq |\{i \mid \xi_i = v_1, 1 \leq i \leq n\}|$$

and, similarly,

$$v_2 \in \mathbb{S}_n \Leftrightarrow M_n(v_2) \leq M_n(v_1) \Leftrightarrow |\{i \mid \xi_i = v_1, 1 \leq i \leq n\}| \leq |\{i \mid \xi_i = v_2, 1 \leq i \leq n\}|.$$

Let

$$R(n) := nM_n(v_1) - nM_n(v_2) = |\{i \mid \xi_i = v_2, 1 \leq i \leq n\}| - |\{i \mid \xi_i = v_1, 1 \leq i \leq n\}|.$$

Observe that we can think of $R(n)$ as a simple symmetric random walk on \mathbb{Z} starting at 0 and

$$\begin{cases} R(n+1) = R(n) - 1, & \text{if } \xi_{n+1} = v_1; \\ R(n+1) = R(n) + 1, & \text{if } \xi_{n+1} = v_2. \end{cases}$$

In other words, $R(0) = 0$, $R(n) = \sum_{i=1}^{\infty} \zeta_i$ where ζ_1, ζ_2, \dots are i.i.d. random variables such that

$$\zeta_1 = \begin{cases} 1, & \text{with probability } 1/2; \\ -1, & \text{with probability } 1/2. \end{cases}$$

Moreover, we have

$$\{v_1 \in \mathbb{S}_n\} = \{M_n(v_1) \leq M_n(v_2)\} = \{R(n) \leq 0\}$$

and

$$\{v_2 \in \mathbb{S}_n\} = \{M_n(v_1) \geq M_n(v_2)\} = \{R(n) \geq 0\}.$$

Now, since a simple symmetric random walk on \mathbb{Z} is recurrent (see [261]), it follows that for every $i = 1, 2$, $\mathbf{P}\{v_i \in \mathbb{S}_n, \text{ i.o.}\} = 1$. Hence, almost always we have

$$\limsup_{n \rightarrow \infty} \mathbb{S}_n = \{v_1, v_2\} = \mathbb{E}\xi,$$

$$\liminf_{n \rightarrow \infty} \mathbb{S}_n = \emptyset,$$

and $\lim_{n \rightarrow \infty} \mathbb{S}_n$ does not exist. \square

A.3.1. Separation and inclusion lemmas.

LEMMA A.3.3. *Let Γ be a locally-finite connected graph, $v \in V(\Gamma)$, and $\{\xi_i\}_{i=1}^{\infty}$ a sequence of i.i.d. random Γ -elements such that $M_{\xi_1}(v)$ is defined. Then*

$$(88) \quad \mathbf{P}\left(M_n(v) \rightarrow M(v) \text{ as } n \rightarrow \infty\right) = 1.$$

Proof. For every $v \in V(\Gamma)$, $M(v)$ is the expectation of the random variable $d^2(v, \xi_1)$. The result follows by the strong law of large numbers applied to $\{d^2(v, \xi_i)\}_{i=1}^{\infty}$. \blacksquare

It is important to notice that in general the convergence in Lemma A.3.3 is not uniform in a sense that, for some distribution μ on a locally finite (infinite) graph Γ and some $\varepsilon > 0$, it is possible that

$$\mathbf{P}\left(\exists N \text{ s.t. } \forall n > N \ \forall v \in V(\Gamma), |M_n(v) - M(v)| < \varepsilon\right) < 1.$$

In other words, the convergence for every vertex, as in Lemma A.3.3, is insufficient to prove the strong law of large numbers, stated in introduction. The next lemma is a key tool in the proof of our strong law of large numbers.

LEMMA A.3.4 (Separation Lemma). *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^{\infty}$ a sequence of i.i.d. random Γ -elements. If the weight function M_{ξ_1} is totally defined, then*

$$\mathbf{P}\left(\exists N \text{ s.t. } \forall n > N, \max_{v \in \mathbb{E}(\xi_1)} M_n(v) < \inf_{u \in V(\Gamma) \setminus \mathbb{E}(\xi_1)} M_n(u)\right) = 1.$$

Proof. Our goal is to prove that for some $\delta > 0$,

$$(89) \quad \mathbf{P}\left(\exists N \ \forall n > N \ \forall v \in \mathbb{E}(\xi_1), \ \forall u \in V(\Gamma) \setminus \mathbb{E}(\xi_1), M_n(u) - M_n(v) \geq \delta\right) = 1.$$

We prove the formula above in two stages. In the first stage we show that for some fixed $v_0 \in \mathbb{E}(\xi_1)$ and for sufficiently large number $m > 0$ the following holds:

(90)

$$\mathbf{P}\left(\exists N \text{ s.t. } \forall n > N \ \forall v \in \mathbb{E}(\xi_1), \ \forall u \in V(\Gamma) \setminus B_{v_0}(m), M_n(u) - M_n(v) \geq \delta\right) = 1$$

in the notation of (85). In the second stage we prove that

$$(91) \quad \mathbf{P}\left(\exists N \text{ s.t. } \forall n > N \ \forall v \in \mathbb{E}(\xi_1), \ \forall u \in B_{v_0}(m) \setminus \mathbb{E}(\xi_1), \ M_n(u) - M_n(v) \geq \delta\right) = 1.$$

Having the formulae above proved we immediately deduce that (89) holds using σ -additivity of measure.

Let $v_0 \in \mathbb{E}(\xi_1)$ and μ be the probability measure on Γ induced by ξ_1 . Since the weight function M is defined at v_0 , we can choose $r \in \mathbb{R}$ as in Lemma A.2.4, such that $\frac{1}{2}M(v_0) \leq \sum_{i \in B_{v_0}(r)} d^2(v_0, i)\mu(i)$. Put $m = 3r$. In Lemma A.2.4 we proved that, if a vertex u is such that $d(u, v_0) \geq 3r$, then

$$(92) \quad M(u) = \sum_{i \in V(\Gamma)} d^2(u, i)\mu(i) \geq 4 \sum_{i \in B_{v_0}(r)} d^2(u, i)\mu(i) \geq 2M(v_0).$$

It implies that $\mathbb{E}(\xi_1) \subseteq B_{v_0}(3r)$.

Since Γ is locally finite, the set $B_{v_0}(r)$ of (85) is finite. We also know from the SLLN for the relative frequencies $\mu_n(u)$ that $\mu_n(u) \xrightarrow{a.s.} \mu(u)$ as $n \rightarrow \infty$. These facts imply that for any $\varepsilon > 0$, the event

$$(93) \quad C_\varepsilon = \{\exists N = N(\varepsilon), \ \forall n > N, \ \forall u \in B_{v_0}(r), \ |\mu_n(u) - \mu(u)| < \varepsilon\}$$

has probability one. In particular, this is true for $\varepsilon = \varepsilon^* = \frac{1}{4} \min\{\mu(u) \mid u \in B_{v_0}(r), \ \mu(u) \neq 0\}$, and in the event C_{ε^*} is a subset of

$$(94) \quad \left\{ \exists N = N(\varepsilon^*), \ \forall n > N, \ \forall u \in V(\Gamma) \setminus B_{v_0}(3r), \ M_n(u) \geq \frac{3}{2}M(v_0) \right\}.$$

Indeed, on the event C_{ε^*} , as in (93), we have $\mu_n(i) \geq \frac{3}{4}\mu(i)$, $i \in B_{v_0}(r)$. Using this fact together with (92), we can write

$$\begin{aligned} M_n(u) &= \sum_{i \in V(\Gamma)} d^2(u, i)\mu_n(i) \geq 4 \sum_{i \in B_{v_0}(r)} d^2(u, i)\mu_n(i) \\ &\geq 3 \sum_{i \in B_{v_0}(r)} d^2(u, i)\mu(i) \geq \frac{3}{2}M(v_0). \end{aligned}$$

Thus we have

$$(95) \quad \mathbf{P}\left(\exists N \text{ s.t. } \forall n > N, \ \forall u \in V(\Gamma) \setminus B_{v_0}(3r), \ M_n(u) \geq \frac{3}{2}M(v_0)\right) = 1.$$

By Lemma A.3.3, for any $v \in V(\Gamma)$ and any $\varepsilon > 0$, we have

$$\mathbf{P}\left(\exists N = N(\varepsilon), \ \forall n > N, \ |M_n(v) - M(v)| < \varepsilon\right) = 1$$

and, since $B_{v_0}(3r)$ is a finite set, we have simultaneous convergence for all vertices in $B_{v_0}(3r)$, i.e.,

$$(96) \quad \mathbf{P}\left(\exists N = N(\varepsilon), \ \forall n > N, \ \forall v \in B_{v_0}(3r), \ |M_n(v) - M(v)| < \varepsilon\right) = 1.$$

In particular, remembering that $\mathbb{E}(\xi_1) \subseteq B_{v_0}(3r)$, for $\varepsilon = M(v_0)/4$,

$$(97) \quad \mathbf{P}\left(\exists N = N(\varepsilon), \ \forall n > N, \ \forall v \in \mathbb{E}(\xi_1), \ \frac{3}{4}M(v) < M_n(v) < \frac{5}{4}M(v)\right) = 1.$$

Finally, we notice that on the intersection of the events in (95) and (97), we have

$$M_n(u) - M_n(v) \geq \frac{3}{2}M(v) - \frac{5}{4}M(v) = \frac{1}{4}M(v) = \frac{1}{4}M(v_0),$$

by virtue of the fact that $M(v_0) = M(v)$ (as both $v_0, v \in \mathbb{E}(\xi_1)$), and formula (90) holds for any δ such that $\delta \leq \frac{1}{4}M(v_0)$.

For the second part of our proof we use statement (96) that holds, in particular, for

$$\varepsilon = \varepsilon' = \frac{1}{4} \min\{M(u) - M(v_0) \mid u \in B_{v_0}(3r), M(u) - M(v_0) > 0\}.$$

It means that, with probability 1, there exists the $N = N(\varepsilon')$ such that for any $n > N$ and all $u \in B_{v_0}(3r)$, we have $|M_n(u) - M(u)| < \varepsilon'$. Moreover, since $\mathbb{E}(\xi_1) \subseteq B_{v_0}(3r)$, we can assert the same for any $v \in \mathbb{E}(\xi_1)$; namely, $|M_n(v) - M(v)| < \varepsilon'$. Together with the fact that $M(u) - M(v_0) > 0$, the obtained inequalities imply that, with probability 1, there exists number $N = N(\varepsilon')$ such that for any $n > N$ and all $u \in B_{v_0}(3r) \setminus \mathbb{E}(\xi_1)$,

$$M_n(v_0) < M(v_0) + \varepsilon' \leq M(v_0) + \frac{1}{4}(M(u) - M(v_0))$$

$$M(u) - \frac{1}{4}(M(u) - M(v_0)) \leq M(u) - \varepsilon' < M_n(u),$$

and, hence,

$$\begin{aligned} M_n(u) - M_n(v_0) &\geq M(u) - \frac{1}{4}(M(u) - M(v_0)) - M(v_0) - \frac{1}{4}(M(u) - M(v_0)) \\ &= \frac{1}{2}(M(u) - M(v_0)) \geq 2\varepsilon', \text{ i.e.,} \end{aligned}$$

$$\mathbf{P}\left(\exists N = N(\varepsilon), \forall n > N, \forall u \in B_{v_0}(3r) \setminus \mathbb{E}(\xi_1) : M_n(u) - M_n(v_0) \geq 2\varepsilon'\right) = 1.$$

Therefore, (91) holds for any $\delta \leq 2\varepsilon'$. Choosing $\delta = \min(\frac{1}{4}M(v_0), 2\varepsilon')$ finishes the proof. ■

COROLLARY A.3.5 (Inclusion Lemma). *Let Γ be a locally-finite connected graph, $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements, and $\mu = \mu_{\xi_1}$. Suppose that the weight function M_ξ is totally defined. Then*

$$\mathbf{P}\left(\limsup_{n \rightarrow \infty} \mathbb{S}(\xi_1, \dots, \xi_n) \subseteq \mathbb{E}(\xi_1)\right) = 1.$$

Proof. Lemma A.3.4 implies that $\mathbf{P}\left(u \notin \limsup \mathbb{S}_n, \text{ for every } u \in V(\Gamma) \setminus \mathbb{E}(\xi_1)\right) = 1$. ■

THEOREM A.3.6 (SLLN for graph-valued random elements with a singleton mean-set). *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements. If the weight function M_{ξ_1} is totally defined and $\mathbb{E}(\xi_1) = \{v\}$ for some $v \in V(\Gamma)$, then*

$$\lim_{n \rightarrow \infty} \mathbb{S}(\xi_1, \dots, \xi_n) = \mathbb{E}(\xi_1)$$

almost surely (with probability one).

Proof. $\mathbf{P}\left(\exists N \text{ s.t. } \forall n > N, M_n(v) < \inf_{u \in V(\Gamma) \setminus \{v\}} M_n(u)\right) = 1$, by Lemma A.3.4, and, hence, $\mathbf{P}\left(\exists N \text{ s.t. } \forall n > N, \mathbb{S}(\xi_1, \dots, \xi_n) = \{v\}\right) = 1$. ■

A.3.2. Case of multi-vertex mean-sets. In this section we investigate a multi-vertex mean-set case and conditions under which the strong law of large numbers holds for such set. We reduce this problem to the question of recurrence of a certain subset in \mathbb{Z}^n relative to a random walk on this integer lattice. If $2 \leq |\mathbb{E}(\xi)| \leq 3$, no restrictive assumptions are required; we formulate and prove the law for these special instances separately. The case $|\mathbb{E}(\xi)| > 3$ requires more technical assumptions, and, thus, more work to handle it.

A.3.2.1. Preliminaries. Assume $\mathbb{E}(\xi) = \{v_1, v_2, \dots, v_k\}$. Our goal is to find conditions that would guarantee the inclusion $\mathbb{E}(\xi_1) \subseteq \limsup_{n \rightarrow \infty} \mathbb{S}_n$ or, without loss of generality, conditions for $v_1 \in \limsup_{n \rightarrow \infty} \mathbb{S}_n$.

By Lemma A.3.4, it follows that, with probability one, for a sequence of random Γ -elements $\{\xi_i\}_{i=1}^{\infty}$, there exists a number N such that for any $n > N$ we have

$$\max\{M_n(v_1), M_n(v_2), \dots, M_n(v_k)\} < \inf_{u \in \Gamma \setminus \{v_1, v_2, \dots, v_k\}} M_n(u).$$

Hence, for any $n > N$, $v_1 \in \mathbb{S}_n$ if and only if $M_n(v_1) \leq M_n(v_i)$ for every $i = 2, \dots, k$. Thus, to achieve our goal, we need to show that the system of inequalities

$$\begin{cases} M_n(v_2) - M_n(v_1) \geq 0, \\ \dots \\ M_n(v_k) - M_n(v_1) \geq 0, \end{cases}$$

is satisfied for infinitely many $n \in \mathbb{N}$.

For $i = 1, \dots, k-1$ and $n \in \mathbb{N}$, define

$$\begin{aligned} R_i(n) &= n(M_n(v_{i+1}) - M_n(v_1)) \\ &= \sum_{s \in \Gamma} (d^2(v_{i+1}, s) - d^2(v_1, s)) \cdot |\{i \mid \xi_i = s, 1 \leq i \leq n\}| \end{aligned}$$

and observe that

$$(98) \quad R_i(n+1) - R_i(n) = \sum_{s \in \Gamma} [d^2(v_{i+1}, s) - d^2(v_1, s)] \mathbf{1}_{\{\xi_{n+1}=s\}},$$

i.e., every $R_i(n)$ represents a random walk on \mathbb{Z} starting at 0. Consider a random walk \bar{R} , associated with v_1 , in \mathbb{Z}^{k-1} , starting at the origin $(0, \dots, 0)$ with the position of the walk after n steps given by

$$\bar{R}(n) = (R_1(n), R_2(n), \dots, R_{k-1}(n)).$$

An increment step for \bar{R} is defined by a vector $\bar{\zeta}(s) = (\zeta_1(s), \dots, \zeta_{k-1}(s))$, $s \in V(\Gamma)$, with probability $\mu(s)$, where

$$\zeta_i(s) = d^2(v_{i+1}, s) - d^2(v_1, s).$$

The following lemma shows the significance of this random walk.

LEMMA A.3.7. *In the notation of this section, $v_1 \in \limsup_{n \rightarrow \infty} \mathbb{S}_n$ if and only if the random walk \bar{R} visits the set $\mathbb{Z}_+^{k-1} = \{(a_1, \dots, a_{k-1}) \mid a_i \geq 0\}$ infinitely often. Therefore,*

$$\mathbf{P}(v_1 \in \limsup_{n \rightarrow \infty} \mathbb{S}_n) = \mathbf{P}(\bar{R}(n) \in \mathbb{Z}_+^{k-1}, \text{ i.o.}).$$

Proof. Follows from the discussion preceding the lemma. ■

It is worth redefining \bar{R} in the terms of transition probability function, as in [261]. Let $\bar{0} \in \mathbb{Z}^{k-1}$ be the zero vector and $x_i = \zeta_i(s)$, $s \in V(\Gamma)$. For every $\bar{x} = (x_1, \dots, x_{k-1}) \in \mathbb{Z}^{k-1}$, we define a function $P(\bar{0}, \bar{x})$ by

$$(99) \quad P(\bar{0}, \bar{x}) = \mu\{s \mid x_i = d^2(v_{i+1}, s) - d^2(v_1, s) \text{ for every } i = 1, \dots, k-1\}.$$

It is trivial to check that this is, indeed, the transition probability for \bar{R} . To continue further, we investigate some properties of our random walk \bar{R} .

LEMMA A.3.8. *Let \bar{R} be a random walk defined above. Then*

$$m_1 = \sum_{\bar{x} \in \mathbb{Z}^{k-1}} \bar{x} P(\bar{0}, \bar{x}) = \bar{0} \quad \text{and} \quad m_2 = \sum_{\bar{x} \in \mathbb{Z}^{k-1}} |\bar{x}|^2 P(\bar{0}, \bar{x}) < \infty.$$

Proof. The first equality trivially holds. For the second one, we get

$$\begin{aligned} \sum_{\bar{x} \in \mathbb{Z}^{k-1}} |\bar{x}|^2 P(\bar{0}, \bar{x}) &= \sum_{s \in V(\Gamma)} \sum_{i=1}^{k-1} \left(d^2(v_{i+1}, s) - d^2(v_1, s) \right)^2 \mu(s) \\ &\leq \sum_{i=1}^{k-1} d^2(v_1, v_{i+1}) \sum_{s \in V(\Gamma)} \left(d(v_1, s) + d(v_{i+1}, s) \right)^2 \mu(s) \\ &\leq \sum_{i=1}^{k-1} d^2(v_1, v_{i+1})(4M(v_1) + 4M(v_{i+1})) < \infty. \end{aligned}$$

■

Clearly, conditions under which this random walk is recurrent would guarantee that $v_1 \subseteq \limsup_{n \rightarrow \infty} \mathbb{S}_n$ (see [261, page 30, Proposition 3.3]). A general (not simple, not symmetric) one-dimensional random walk is recurrent if its first moment is zero and its first absolute moment is finite (see [261], pg. 23). Sufficient conditions for the recurrence of two-dimensional random walk involve the finiteness of its second moment and can be found in [261, page 83]. The result stated there indicates that genuinely 2-dimensional random walk is recurrent if its first moment is zero, and its second moment is finite. Let us recall some important notions before we go on.

Consider an arbitrary random walk \bar{R} on \mathbb{Z}^n given by a transition probability P , as in (99). The support, $\text{supp}(P)$, of the probability measure P is defined to be the set

$$\text{supp}(P) = \{\bar{v} \in \mathbb{Z}^n \mid P(\bar{v}) \neq 0\}$$

of all possible one-step increments of \bar{R} . Further, with \bar{R} , one can associate an abelian subgroup $A_{\bar{R}}$ of \mathbb{Z}^n generated by the vectors in $\text{supp}(P)$. It is well-known in group theory that any subgroup $A_{\bar{R}}$ of \mathbb{Z}^n is isomorphic to \mathbb{Z}^k , where $k \leq n$ (the reader can also check [261, Proposition 7.1 on pg. 65] for details), in which case we write $\dim(A_{\bar{R}}) = k$ and say that \bar{R} is *genuinely k-dimensional*. Let us stress that we speak of an n -dimensional random walk on \mathbb{Z}^n when $P(0, \bar{x})$ is defined for all \bar{x} in \mathbb{Z}^n ; this walk is genuinely n -dimensional if $\dim(A_{\bar{R}}) = n$. We say that \bar{R} is *aperiodic* if $A_{\bar{R}} = \mathbb{Z}^n$. Observe that genuinely n -dimensional random walk does not have to be aperiodic. A standard simple random walk, which we denote by $S = S(n)$, is an example of an aperiodic random walk on \mathbb{Z}^n .

It will be convenient for our presentation to define a vector space $V_{\bar{R}} \subset \mathbb{R}^n$ spanned by the vectors in $\text{supp}(P)$. It is easy to see that the genuine dimension of \bar{R}

is equal to the dimension of $V_{\overline{R}}$. We shall need another notion for our developments. Assume that D is an $k \times n$ matrix (not necessarily integer valued) which maps $A_{\overline{R}}$ onto \mathbb{Z}^k . Then D naturally induces a random walk \overline{R}^D on \mathbb{Z}^k with transition probability P^D given by

$$P^D(\overline{u}) = P(\overline{v} \in \mathbb{Z}^n \mid D(\overline{v}) = \overline{u})$$

for every $\overline{u} \in \mathbb{Z}^k$.

A.3.2.2. Strong law of large numbers for two or three vertices mean-sets. Now, we can easily prove our strong law of large numbers for mean-sets with two or three elements.

THEOREM A.3.9 (SLLN for graph random elements with two or three point mean-set). *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^{\infty}$ a sequence of i.i.d. random Γ -elements. If the weight function M_{ξ_1} is totally defined and $2 \leq |\mathbb{E}(\xi)| \leq 3$, then*

$$\limsup_{n \rightarrow \infty} \mathbb{S}_n = \mathbb{E}(\xi_1)$$

holds with probability one.

Proof. Assume that $\mathbb{E}(\xi) = \{v_1, v_2\}$. Then the random walk \overline{R} is one-dimensional. It is recurrent if

$$\sum_{s \in \Gamma} |\zeta_1(s)|\mu(s) < \infty \text{ and } \sum_{s \in \Gamma} \zeta_1(s)\mu(s) = 0$$

(see [261], p. 23). The equality $M(v_1) = M(v_2)$ implies the second conditions and

$$\begin{aligned} \sum_{s \in \Gamma} |\zeta_1(s)|\mu(s) &= \sum_{s \in \Gamma} |d^2(v_2, s) - d^2(v_1, s)|\mu(s) \\ &\leq \sum_{s \in \Gamma} (d^2(v_2, s) + d^2(v_1, s))\mu(s) = M(v_1) + M(v_2) < \infty \end{aligned}$$

implies the first condition. Hence, \overline{R} is recurrent, and takes on positive and negative values infinitely often. We conclude that almost always $\limsup_{n \rightarrow \infty} \mathbb{S}_n = \{v_1, v_2\} = \mathbb{E}\xi$.

Assume that $\mathbb{E}(\xi) = \{v_1, v_2, v_3\}$. Then the random walk \overline{R} can be genuinely 0, 1, or 2-dimensional. The first case is trivial, the second can be considered as the case when $|\mathbb{E}(\xi)| = 2$. So, assume \overline{R} is genuinely 2-dimensional. By Lemma A.3.8, the first moment of \overline{R} is $(0, 0)$ and the second moment is finite. Now, it follows from [261, Theorem 8.1] that \overline{R} is recurrent. In particular, \mathbb{Z}_+^{k-1} is visited infinitely often with probability 1.

In both cases, it follows from Lemma A.3.7 that $\mathbf{P}(v_1 \in \limsup_{n \rightarrow \infty} \mathbb{S}_n) = 1$. Hence the result. ■

A.3.2.3. Case of mean-sets with more than three vertices. Recall that a subset of \mathbb{Z}^n is called *recurrent* if it is visited by a given random walk infinitely often with probability one, and it is *transient* otherwise. A criterion for recurrence of a set for a simple random walk was obtained in [141] for $n = 3$ (it can also be found in [261, Theorem 26.1]). It turns out that the criterion does not depend on a random walk in question. This is the subject of the extension of the Wiener's test, proved in [273], that we state below. This invariance principle is one of the main tools we use in our investigation of the recurrence properties of the positive octant in \mathbb{Z}^n for \overline{R} .

THEOREM (Extension of Wiener's test, [273]). *Let $n \geq 3$. An infinite subset A of \mathbb{Z}^n is either recurrent for each aperiodic random walk \bar{R} on \mathbb{Z}^n with mean zero and a finite variance, or transient for each of such random walks.*

For a positive constant $\alpha \in \mathbb{R}$ and a positive integer $m \leq n$ define a subset of \mathbb{R}^n

$$\begin{aligned} Cone_\alpha^m = \{(x_1, \dots, x_n) \in \mathbb{R}^n \mid &x_1 = 0, \dots, x_{n-m} = 0, \\ &\sqrt{x_{n-m+1}^2 + \dots + x_{n-1}^2} \leq \alpha x_n\} \end{aligned}$$

called an m -dimensional cone in \mathbb{R}^n . If $m = n$, then we omit the superscript in $Cone_\alpha^m$. For an $n \times n$ matrix D and a set $A \subseteq \mathbb{R}^n$, define a set $A^D = \{D \cdot \bar{v} \mid \bar{v} \in A\}$, which is a linear transformation of A . If D is an orthogonal matrix, then the set $(Cone_\alpha)^D$ is called a *rotated cone*. As in [141], for any non-decreasing function $i : \mathbb{N} \rightarrow \mathbb{R}_+$ define a set

$$Thorn_i = \{\bar{v} \in \mathbb{Z}^n \mid \sqrt{v_1^2 + \dots + v_{n-1}^2} \leq i(v_n)\}.$$

Observe that $Cone_\alpha \cap \mathbb{Z}^n = Thorn_i$ where $i(t) = \alpha t$.

THEOREM A.3.10. *For any $\alpha > 0$ and any orthogonal matrix D ,*

$$\mathbf{P}(S(n) \in (Cone_\alpha)^D, \text{ i.o.}) = 1,$$

i.e., the probability that the simple random walk on \mathbb{Z}^n visits $(Cone_\alpha)^D$ infinitely often is 1.

Proof. Direct consequence of (6.1) and (4.3) in [141], where the criterion for recurrence of $Thorn_i$ is given. ■

The next two lemmas are obvious

LEMMA A.3.11. *Assume that a set $A \subseteq \mathbb{R}^n$ contains a rotated cone. Then for any invertible $n \times n$ matrix D , the set A^D contains a rotated cone.*

LEMMA A.3.12. *If $S_1 \subseteq S_2 \subseteq \mathbb{R}^n$ and S_1 is visited by the simple random walk infinitely often with probability 1 then S_2 is visited by the simple random walk infinitely often with probability one.*

Now, we return to our strong law of large numbers for multi-vertex mean-sets. Assume that $\mathbb{E}\xi = \{v_1, \dots, v_k\}$, where $k \geq 4$. Let \bar{R}^i be a random walk on \mathbb{Z}^{k-1} , associated with v_i , where $i = 1, \dots, k$ (in our notation, $\bar{R} = \bar{R}^1$). This is a $(k-1)$ -dimensional random walk which, in general, is not aperiodic. In fact, \bar{R}^i is not even genuinely $(k-1)$ -dimensional. Fortunately, it turns out that it does not matter to what vertex v_i we associate our random walk, since the choice of the vertex does not affect the dimension of the corresponding walk, as the following lemma shows.

LEMMA A.3.13. *Let μ be a probability measure on a locally finite graph Γ such that $\mathbb{E}\mu = \{v_1, \dots, v_k\}$, where $k \geq 2$. Then the random walks $\bar{R}^1, \dots, \bar{R}^k$, associated with vertices v_1, \dots, v_k respectively, all have the same genuine dimension.*

Proof. We prove that random walks \bar{R}^1 and \bar{R}^2 have the same genuine dimension. Recall that the subgroup $A_{\bar{R}^1}$ is generated by the set of vectors $\bar{v}^1 \in \mathbb{Z}^{k-1}$ such that for some $s \in supp(\mu)$, $\bar{v}^1 = \bar{v}^1(s) = (d^2(v_2, s) - d^2(v_1, s), d^2(v_3, s) - d^2(v_1, s), \dots, d^2(v_k, s) - d^2(v_1, s))$ and the subgroup $A_{\bar{R}^2}$ is generated by the set

of vectors $\bar{v}^2 \in \mathbb{Z}^{k-1}$ such that for some $s \in \text{supp}(\mu)$, $\bar{v}^2 = \bar{v}^2(s) = (d^2(v_1, s) - d^2(v_2, s), d^2(v_3, s) - d^2(v_2, s), \dots, d^2(v_k, s) - d^2(v_1, s))$. Observe that for every $s \in \text{supp}(\mu)$ the equality $\bar{v}^2(s) = D \cdot \bar{v}^1(s)$ holds, where D is a $(k-1) \times (k-1)$ matrix

$$D = \begin{pmatrix} -1 & 0 & 0 & 0 & \dots \\ -1 & 1 & 0 & 0 & \dots \\ -1 & 0 & 1 & 0 & \dots \\ -1 & 0 & 0 & 1 & \dots \\ \dots \end{pmatrix}.$$

Therefore, $A_{\bar{R}^2} = (A_{\bar{R}^1})^D$. Since the matrix D is invertible it follows that $A_{\bar{R}^1}$ and $A_{\bar{R}^2}$ have the same dimension. ■

THEOREM A.3.14. *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements. Assume that the weight function M is totally defined and $\mathbb{E}(\xi) = \{v_1, \dots, v_k\}$, where $k \geq 4$. If the random walk \bar{R}^1 associated to v_1 is genuinely $(k-1)$ -dimensional, then*

$$\limsup_{n \rightarrow \infty} \mathbb{S}_n = \mathbb{E}(\xi_1)$$

holds with probability one.

Proof. Since \bar{R}^1 is genuinely $(k-1)$ -dimensional it follows that the subgroup $A_{\bar{R}^1}$ is isomorphic to \mathbb{Z}^{k-1} and there exists an invertible matrix D that isomorphically maps $A_{\bar{R}^1} \subseteq \mathbb{Z}^{k-1}$ onto \mathbb{Z}^{k-1} . Consider a set $\mathbb{R}_+^{k-1} = \{(x_1, \dots, x_{k-1}) \mid x_i \geq 0\}$. Obviously, $\mathbf{P}(\bar{R}^1 \in \mathbb{Z}_+^{k-1} \text{ i.o.}) = \mathbf{P}(\bar{R}^1 \in \mathbb{R}_+^{k-1} \text{ i.o.})$.

Let $(\bar{R}^1)^D$ be the random walk on \mathbb{Z}^{k-1} induced by D by application of D to \bar{R}^1 . The random walk $(\bar{R}^1)^D$ is aperiodic since D maps $A_{\bar{R}^1}$ onto \mathbb{Z}^{k-1} and, by construction of $(\bar{R}^1)^D$,

$$\mathbf{P}(\bar{R}^1 \in \mathbb{R}_+^{k-1} \text{ i.o.}) = \mathbf{P}((\bar{R}^1)^D \in (\mathbb{R}_+^{k-1})^D \text{ i.o.}).$$

Let S be the simple random walk on \mathbb{Z}^{k-1} . Since $(\bar{R}^1)^D$ and S are both aperiodic random walks on \mathbb{Z}^{k-1} , it follows from the Invariance Principle (Extension of Wiener's test) that

$$\mathbf{P}((\bar{R}^1)^D \in (\mathbb{R}_+^{k-1})^D \text{ i.o.}) = \mathbf{P}(S \in (\mathbb{R}_+^{k-1})^D \text{ i.o.}).$$

Clearly, the set \mathbb{R}_+^{k-1} contains a rotated cone and, hence, by Lemma A.3.11, its image under an invertible linear transformation D contains a rotated cone too. Now, by Theorem A.3.10, $\mathbf{P}(S \in (\mathbb{R}_+^{k-1})^D \text{ i.o.}) = 1$. Thus, $\mathbf{P}(\bar{R}^1 \in \mathbb{Z}_+^{k-1} \text{ i.o.}) = 1$ and by Lemma A.3.7,

$$\mathbf{P}(v_1 \in \limsup_{n \rightarrow \infty} \mathbb{S}_n) = 1.$$

Finally, it follows from Lemma A.3.13 that for any $i = 2, \dots, k$ the random walk \bar{R}^i is genuinely $(k-1)$ -dimensional. For any $i = 2, \dots, k$ we can use the same argument as for v_1 to prove that $\mathbf{P}(v_i \in \limsup_{n \rightarrow \infty} \mathbb{S}_n) = 1$. Hence the result. ■

A.3.2.4. *The case when random walk is not genuinely $(k-1)$ -dimensional.* The case when \overline{R}^1 is not genuinely $(k-1)$ -dimensional is more complicated. To answer the question whether v_1 belongs to $\limsup_{n \rightarrow \infty} \mathbb{S}_n$ (namely, how often $v_1 \in \mathbb{S}_n$), we need to analyze how the space $V_{\overline{R}^1}$ “sits” in \mathbb{R}^{k-1} . We know that the subgroup $A_{\overline{R}^1} \subset \mathbb{Z}^{k-1}$ is isomorphic to \mathbb{Z}^m , where $m < k-1$ in the case under consideration. Therefore, there exists a $m \times (k-1)$ matrix D which maps the subgroup $A_{\overline{R}^1}$ onto \mathbb{Z}^m and which is injective onto $A_{\overline{R}^1}$. Furthermore, the mapping D maps the subspace $V_{\overline{R}^1}$ bijectively onto \mathbb{R}^m . The linear mapping D induces an aperiodic random walk $(\overline{R}^1)^D$ on \mathbb{Z}^m in a natural way and $\mathbf{P}(\overline{R}^1 \in (\mathbb{R}_+^{k-1}) \text{ i.o.}) = \mathbf{P}(\overline{R}^1 \in (\mathbb{R}_+^{k-1} \cap V_{\overline{R}^1}) \text{ i.o.}) = \mathbf{P}((\overline{R}^1)^D \in (\mathbb{R}_+^{k-1} \cap V_{\overline{R}^1})^D \text{ i.o.})$. The main problem here is to understand the structure of the set $(\mathbb{R}_+^{k-1} \cap V_{\overline{R}^1})^D$ and, to be more precise, the structure of the set $B_{\overline{R}^1} = \mathbb{R}_+^{k-1} \cap V_{\overline{R}^1}$. Clearly $B_{\overline{R}^1}$ is a monoid, i.e., contains the trivial element and a sum of any two elements in $B_{\overline{R}^1}$ belongs to $B_{\overline{R}^1}$. We can define dimension of $B_{\overline{R}^1}$ to be the maximal number of linearly independent vectors in $B_{\overline{R}^1}$.

THEOREM A.3.15. *Suppose $A_{\overline{R}^1} \simeq \mathbb{Z}^m$ and the set $B_{\overline{R}^1}$ has dimension m . Then $\mathbf{P}(v_i \in \limsup_{n \rightarrow \infty} \mathbb{S}_n) = 1$.*

Proof. Since $B_{\overline{R}^1}$ is a monoid of dimension m it is not hard to see that $B_{\overline{R}^1}$ contains an m -dimensional rotated cone. Since D is a linear isomorphism from $V_{\overline{R}^1}$ onto \mathbb{R}^m it follows by Lemma A.3.11 that $(B_{\overline{R}^1})^D$ contains an m -dimensional rotated cone in \mathbb{R}^m . If S is a simple random walk in \mathbb{Z}^m then $\mathbf{P}(S \in (B_{\overline{R}^1})^D \text{ i.o.}) = 1$ and since S and $(\overline{R}^1)^D$ are both aperiodic, by the extension of Wiener’s test (Invariance Principle), we see that $\mathbf{P}((\overline{R}^1)^D \in (B_{\overline{R}^1})^D \text{ i.o.}) = 1$. Hence, $\mathbf{P}(\overline{R}^1 \in (\mathbb{R}_+^{k-1}) \text{ i.o.}) = 1$ and by Lemma A.3.7, $\mathbf{P}(v_i \in \limsup_{n \rightarrow \infty} \mathbb{S}_n) = 1$. ■

Below we investigate under what conditions the subgroup $A_{\overline{R}^1}$ and the set $B_{\overline{R}^1}$ have the same dimension m .

LEMMA A.3.16. *Assume that $A_{\overline{R}^1}$ contains a positive vector. Then $A_{\overline{R}^1}$ and the set $\mathbb{R}_+^{k-1} \cap V_{\overline{R}^1}$ have the same dimension.*

Proof. Straightforward. ■

LEMMA A.3.17. *Assume that $\mu(v_1) \neq 0$. Then $A_{\overline{R}^1}$ and the set $\mathbb{R}_+^{k-1} \cap V_{\overline{R}^1}$ have the same dimension.*

Proof. Observe that if $\mu(v_1) \neq 0$ then $A_{\overline{R}^1}$ contains the vector $(d^2(v_2, v_1), \dots, d^2(v_k, v_1))$ which has all positive coordinates. Therefore, by Lemma A.3.16 the set $A_{\overline{R}^1}$ and $\mathbb{R}_+^{k-1} \cap V_{\overline{R}^1}$ have the same dimension. ■

THEOREM A.3.18. *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements. Assume that the weight function M_{ξ_1} is totally defined and $\mathbb{E}(\xi) = \{v_1, \dots, v_k\}$, where $k \geq 4$. If $\mathbb{E}(\xi_1) \subseteq \text{supp}(\mu)$ then $\limsup_{n \rightarrow \infty} \mathbb{S}_n = \mathbb{E}(\xi_1)$ holds with probability one.*

Proof. Follows from Lemmas A.3.17, A.3.16, and Theorem A.3.15. ■

A.4. Concentration of measure inequalities

Every area of mathematics in general, as well as every trend of probability theory in particular, possesses some important inequalities. Inequalities, as humble as they may seem, often provide necessary bounds on measure concentration and rates of convergence.

Concentration inequalities are upper bounds on the rate of convergence (in probability) of sample (empirical) means to their ensemble counterparts (actual means). Chebyshev inequality and Chernoff-Hoeffding exponential bounds are classical examples of such inequalities in probability theory. In this section, we prove analogues of the classical Chebyshev's inequality and Chernoff-Hoeffding-like bounds — the concentration of measure inequalities for graph-(group-)valued random elements.

A.4.1. Chebyshev's inequality for graphs/groups. The classical Chebyshev's inequality asserts that if ξ is a random variable with $\mathbb{E}(\xi^2) < \infty$, then for any $\varepsilon > 0$, we have

$$(100) \quad \mathbf{P}(|\xi - \mathbb{E}(\xi)| \geq \varepsilon) \leq \frac{\sigma^2}{\varepsilon^2},$$

where $\sigma^2 = \text{Var}(\xi)$; see [19].

Chebyshev discovered it when he was trying to prove the law of large numbers, and the inequality is widely used ever since. Chebyshev's inequality is a result concerning the concentration of measure, giving a quantitative description of this concentration. Indeed, it provides a bound on the probability that a value of a random variable ξ with finite mean and variance will differ from the mean by more than a fixed number ε . In other words, we have a crude estimate for concentration of probabilities around the expectation, and this estimate has a big theoretical significance.

The inequality (100) applied to the sample mean random variable $\bar{X} = \frac{S_n}{n}$, where $S_n = \xi_1 + \dots + \xi_n$, $\mathbb{E}(\xi_i) = m$, $\text{Var}(\xi_i) = \sigma^2$, $i = 1, \dots, n$ results in

$$(101) \quad \mathbf{P}(|\bar{X} - m| \geq \varepsilon) \leq \frac{\sigma^2}{n\varepsilon^2}.$$

The goal is to prove a similar inequality for a graph-valued random element ξ .

LEMMA A.4.1. *Let μ be a distribution on a locally finite graph Γ such that M is defined. If for some $r \in \mathbb{N}$ and $v_0 \in V(\Gamma)$ the inequality*

$$(102) \quad \sum_{s \in V(\Gamma) \setminus B_{v_0}(r/2)} d(v_0, s)\mu(s) - \frac{r}{2}\mu(v_0) < 0$$

holds, then $M(u) > M(v_0)$ for any $u \in V(\Gamma) \setminus B_{v_0}(r)$.

Proof. Indeed, pick any $u \in V(\Gamma) \setminus B_{v_0}(r)$ and put $d = d(v_0, u)$. Then

$$\begin{aligned} M(u) - M(v_0) &= \sum_{s \in V(\Gamma)} (d^2(u, s) - d^2(v_0, s))\mu(s) \\ &\geq d^2\mu(v_0) - \sum_{d(v_0, s) > d(u, s)} (d^2(v_0, s) - d^2(u, s))\mu(s) \\ &\geq d^2\mu(v_0) - 2d \sum_{d(v_0, s) > d(u, s)} d(v_0, s)\mu(s) \\ &\geq d^2\mu(v_0) - 2d \sum_{s \in V(\Gamma) \setminus B_{v_0}(r/2)} d(v_0, s)\mu(s). \end{aligned}$$

Since $d > r$, it follows from the assumption of the lemma that we get a positive quantity at the end. To make it more clear, let us note that in the last estimate, we used the observation that

$$\left\{ s \in V(\Gamma) \mid d(v_0, s) > d(u, s) \right\} \cap B_{v_0}(r/2) = \emptyset,$$

which implies that $\left\{ s \in V(\Gamma) \mid d(v_0, s) > d(u, s) \right\} \subseteq V(\Gamma) \setminus B_{v_0}(r/2)$ and, therefore,

$$\sum_{d(v_0, s) > d(u, s)} d(v_0, s)\mu(s) \leq \sum_{s \in V(\Gamma) \setminus B_{v_0}(r/2)} d(v_0, s)\mu(s).$$

We conclude that $M(u) > M(v_0)$ as required. ■

THEOREM A.4.2. *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements. If the weight function M_{ξ_1} is totally defined and $\mathbb{E}(\xi_1) = \{v\}$ for some $v \in V(\Gamma)$ then there exists a constant $C = C(\Gamma, \xi_1) > 0$ such that*

$$(103) \quad \mathbf{P}\left(\mathbb{S}(\xi_1, \dots, \xi_n) \neq \{v\}\right) \leq \frac{C}{n}.$$

Proof. It follows from the definition of the sample mean-set that

$$\{\mathbb{S}_n \neq \{v\}\} = \{\exists u \in V(\Gamma) \setminus \{v\}, M_n(u) \leq M_n(v)\}.$$

Hence, it is sufficient to prove that $\mathbf{P}\left(\exists u \in V(\Gamma) \setminus \{v\}, M_n(u) \leq M_n(v)\right) \leq \frac{C}{n}$, for some constant C . We do it in two stages. We show that for some $v_0 \in V(\Gamma)$ and constants $r \in \mathbb{N}$, $C_1, C_2 \in \mathbb{R}$ such that $v \in B_{v_0}(r)$ and inequalities

$$(104) \quad \mathbf{P}\left(\exists u \in B_{v_0}(r) \setminus \{v\}, M_n(u) \leq M_n(v)\right) \leq \frac{C_1}{n}$$

and

$$(105) \quad \mathbf{P}\left(\exists u \in V(\Gamma) \setminus B_{v_0}(r), M_n(u) \leq M_n(v_0)\right) \leq \frac{C_2}{n}$$

hold. Clearly, for any $u, v_0, v \in V(\Gamma)$ if $M_n(u) \leq M_n(v)$ then either $M_n(u) \leq M_n(v_0)$ or $M_n(v_0) \leq M_n(v)$. It is not hard to see that if we find C_1 and C_2 satisfying (104) and (105), respectively, then (103) holds for $C = C_1 + C_2$ and the theorem is proved.

First we argue (105). Choose any $v_0 \in V(\Gamma)$ such that $\mu(v_0) > 0$ and $r \in \mathbb{N}$ such that the inequality (102) holds. We can choose such r since $M^{(1)}(v_0)$ is defined. Observe that the left-hand side of the inequality above is the expectation of a

random variable $\eta : V \rightarrow \mathbb{R}$ defined as $\eta(s) = d(v_0, s)\mathbf{1}_{V(\Gamma) \setminus B_{v_0}(r/2)}(s) - \frac{r}{2}\mathbf{1}_{v_0}(s)$, $s \in V(\Gamma)$, where $\mathbf{1}_\cdot(s)$ is an indicator function:

$$\mathbf{1}_A(s) = \begin{cases} 1, & \text{if } s \in A, \\ 0, & \text{if } s \notin A, \end{cases}$$

with $A \subseteq V(\Gamma)$ and $s \in V(\Gamma)$. Since by our assumption $M \equiv M^{(2)}$ is defined, it follows that $\sigma^2(\eta)$ is defined, and, applying Lemma A.4.1 and the Chebyshev inequality with $\varepsilon = |\mathbb{E}\eta|/2$, we obtain

$$\begin{aligned} & \mathbf{P}\left(\exists u \in V(\Gamma) \setminus B_{v_0}(r), M_n(u) \leq M_n(v_0)\right) \\ & \leq \mathbf{P}\left(\left|\sum_{s \in V(\Gamma) \setminus B_{v_0}(r/2)} d(v_0, s)\mu_n(s) - \frac{r}{2}\mu_n(v_0) - \mathbb{E}\eta\right| \geq |\mathbb{E}\eta|/2\right) \leq \frac{4\sigma^2(\eta)}{n|\mathbb{E}\eta|^2}. \end{aligned}$$

Hence, inequality (105) holds for $C_2 = C_2(r, v_0, \mu) = \frac{4\sigma^2(\eta)}{|\mathbb{E}\eta|^2}$. To prove (104) we notice that for any $u \in V(\Gamma) \setminus \{v\}$,

$$M(u) - M(v) = \sum_{s \in V(\Gamma)} (d(u, s) - d(v, s))(d(u, s) + d(v, s))\mu(s),$$

i.e., $M(u) - M(v)$ is the expectation of a random variable $\tau : V \rightarrow \mathbb{R}$ defined as

$$\tau_{u,v}(s) = (d(u, s) - d(v, s))(d(u, s) + d(v, s)), \quad s \in V(\Gamma).$$

Furthermore, since M_{ξ_1} is defined and for every $s \in V(\Gamma)$, $d(u, s) - d(v, s) \leq d(v, u)$, it is easy to see that $\sigma^2(\tau_{u,v}(s)) < \infty$. Thus, by the Chebyshev inequality for the sample average of $\tau_{u,v}(s)$, the following holds:

$$\mathbf{P}\left(|M_n(u) - M_n(v) - (M(u) - M(v))| \geq \varepsilon\right) \leq \frac{\sigma^2(\tau_{u,v}(s))}{n\varepsilon^2}.$$

Now, if $0 < \varepsilon < M(u) - M(v)$, then

$$\mathbf{P}\left(M_n(u) < M_n(v)\right) \leq \mathbf{P}\left(|M_n(u) - M_n(v) - (M(u) - M(v))| \geq \varepsilon\right).$$

Finally, we choose ε to be $\frac{1}{2}\inf\{M(u) - M(v) \mid u \in B_{v_0}(r) \setminus \{v\}\}$ and using σ -additivity of measure we see that inequality (104) holds for the constant $C_1 = \varepsilon^{-2} \sum_{u \in B_{v_0}(r)} \sigma^2(\tau_{u,v}(s))$. ■

In fact, one can easily generalize the previous theorem to the following statement.

THEOREM A.4.3. *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements. If the weight function M_{ξ_1} is totally defined then there exists a constant $C = C(\Gamma, \xi_1) > 0$ such that*

$$(106) \quad \mathbf{P}\left(\mathbb{S}(\xi_1, \dots, \xi_n) \not\subseteq \mathbb{E}(\xi)\right) \leq \frac{C}{n}.$$

A.4.2. Chernoff-Hoeffding-like bound for graphs/groups. Let x_i be independent random variables. Assume that each x_i is almost surely bounded, i.e., assume that for every $i \in \mathbb{N}$ there exists $a_i, b_i \in \mathbb{R}$ such that $\mathbf{P}(x_i - \mathbb{E}x_i \in [a_i, b_i]) = 1$. Then for $S_n = \sum_{i=1}^n x_i$ and for any $\varepsilon > 0$ we have the inequality (called the Hoeffding's inequality)

$$\mathbf{P}(|S_n - \mathbb{E}S_n| \geq n\varepsilon) \leq 2 \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

If x_i are identically distributed then we get the inequality

$$\mathbf{P}\left(\left|\frac{1}{n}(x_1 + \dots + x_n) - \mathbb{E}x_1\right| \geq \varepsilon\right) \leq 2 \exp\left(-\frac{2\varepsilon^2}{(b-a)^2}n\right).$$

Techniques of the previous section can be used to find a similar bound on $\mathbf{P}(\mathbb{S}(\xi_1, \dots, \xi_n) \not\subseteq \mathbb{E}(\xi))$ for a sequence of iid graph-valued ξ_i satisfying some additional assumptions.

THEOREM A.4.4. *Let Γ be a locally-finite connected graph and $\{\xi_i\}_{i=1}^\infty$ a sequence of i.i.d. random Γ -elements. If the weight function $M_{\xi_1}(\cdot)$ is totally defined and μ_{ξ_1} has finite support then for some constant $C > 0$,*

$$(107) \quad \mathbf{P}\left(\mathbb{S}(\xi_1, \dots, \xi_n) \not\subseteq \mathbb{E}(\xi)\right) \leq O(e^{-Cn}).$$

Proof. Similar to the proof of Theorem A.4.2. We find $v_0 \in V(\Gamma)$, $r \in \mathbb{N}$, and constants $C_1, C_2 > 0$ such that inequalities

$$(108) \quad \mathbf{P}\left(\exists u \in B_{v_0}(r) \setminus \{v\}, M_n(u) \leq M_n(v)\right) \leq O(e^{-C_1 n})$$

and

$$(109) \quad \mathbf{P}\left(\exists u \in V(\Gamma) \setminus B_{v_0}(r), M_n(u) \leq M_n(v_0)\right) \leq O(e^{-C_2 n})$$

hold.

Choose $v_0 \in V(\Gamma)$ and $r \in \mathbb{N}$ exactly the same way as in Theorem A.4.2. Note that a random variable $\eta(s) = d(v_0, s)\mathbf{1}_{V(\Gamma) \setminus B_{v_0}(r/2)}(s) - \frac{r}{2}\mathbf{1}_{v_0}(s)$ (where $s \in V(\Gamma)$) is almost surely bounded. Choose a lower and an upper bounds for η and denote them by a and b , respectively. Now, applying Hoeffding's inequality to η with $\varepsilon = |\mathbb{E}\eta|/2$ we obtain

$$\begin{aligned} & \mathbf{P}\left(\exists u \in V(\Gamma) \setminus B_{v_0}(r), M_n(u) \leq M_n(v_0)\right) \\ & \leq \mathbf{P}\left(\left|\sum_{s \in V(\Gamma) \setminus B_{v_0}(r/2)} d(v_0, s)\mu_n(s) - \frac{r}{2}\mu_n(v_0) - \mathbb{E}\eta\right| \geq |\mathbb{E}\eta|/2\right) \\ & \leq 2 \exp\left(-\frac{|\mathbb{E}\eta|^2}{2(b-a)^2}n\right). \end{aligned}$$

Therefore, (109) holds for $C_2 = \frac{|\mathbb{E}\eta|^2}{2(b-a)^2}$.

To prove (108) we notice that for any $v \in \mathbb{E}(\xi)$ and $u \in V(\Gamma) \setminus \mathbb{E}(\xi)$ we have

$$M(u) - M(v) = \sum_{s \in V(\Gamma)} (d(u, s) - d(v, s))(d(u, s) + d(v, s))\mu(s),$$

i.e., $M(u) - M(v)$ is the expectation of a random variable $\tau_{u,v} : V \rightarrow \mathbb{R}$ defined as

$$\tau_{u,v}(s) = (d(u, s) - d(v, s))(d(u, s) + d(v, s)), \quad s \in V(\Gamma).$$

Furthermore, since ξ_1 has finite support it follows that the random variable $\tau_{u,v}(s)$ almost surely belongs to $[a_{u,v}, b_{u,v}]$. Thus, by the Hoeffding's inequality for the sample average of $\tau_{u,v}(s)$, the following holds:

$$\mathbf{P}\left(|M_n(u) - M_n(v) - (M(u) - M(v))| \geq \varepsilon\right) \leq 2 \exp\left(-\frac{2\varepsilon^2}{(b_{u,v} - a_{u,v})^2}n\right).$$

Now, if $0 < \varepsilon < M(u) - M(v)$, then

$$\mathbf{P}\left(M_n(u) < M_n(v)\right) \leq \mathbf{P}\left(|M_n(u) - M_n(v) - (M(u) - M(v))| \geq \varepsilon\right).$$

Choose ε to be $\frac{1}{2}\inf\{M(u) - M(v) \mid v \in \mathbb{E}(\xi), u \in B_{v_0}(r) \setminus \mathbb{E}(\xi)\}$ and $\delta = \max\{b_{u,v} - a_{u,v} \mid v \in \mathbb{E}(\xi), u \in B_{v_0}(r) \setminus \mathbb{E}(\xi)\}$. Finally, using σ -additivity of measure we see that inequality (108) holds for the constant $C_1 = \frac{2\varepsilon^2}{\delta^2}$. ■

A.5. Configurations of mean-sets

In this section, we discuss configurations of mean-sets on graphs and, in particular, on trees and free groups. It is easy to observe that for every connected graph Γ and $v \in V(\Gamma)$ there exists a measure μ such that $\mathbb{E}(\mu) = \{v\}$. On the other hand, it is easy to see that not any subset of $V(\Gamma)$ can be realized as $\mathbb{E}(\mu)$. For instance, consider a graph as in Figure A.2. Let $\mu_0 = \mu(v_0)$, $\mu_1 = \mu(v_1)$, $\mu_2 = \mu(v_2)$, $M_0 = M(v_0)$, $M_1 = M(v_1)$, $M_2 = M(v_2)$. Then $M_1 = \mu_0 + 4\mu_2$, $M_0 = \mu_1 + \mu_2$, $M_2 = 4\mu_1 + \mu_0$. Clearly, for no values of μ_0 , μ_1 , and μ_2 both inequalities $M_0 > M_1$ and $M_0 > M_2$ can hold simultaneously (since we cannot have $2M_0 > M_1 + M_2$). Thus, v_1 and v_2 cannot comprise $\mathbb{E}\mu$. In fact, a tree can have only a limited configuration of centers as proved in Proposition A.5.7 below.

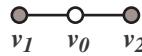


FIGURE A.2. Impossible configuration of centers (gray vertices).

Let Γ be a graph. We say that $v_0 \in V(\Gamma)$ is a *cut-point* if removing v_0 from Γ results in a disconnected graph. The same definition holds for any metric space. It turns out that existence of a *cut-point* in Γ affects configurations of mean-sets. The following lemma provides a useful inequality that holds for any metric space with a cut-point.

LEMMA A.5.1 (Cut-point inequality). *Let (Γ, d) be a metric space and v_0 a cut point in Γ . If v_1, v_2 belong to distinct connected components of $\Gamma \setminus \{v_0\}$ then for any $s \in V(\Gamma)$ the inequality*

$$(110) \quad d(v_0, v_2)(d^2(v_1, s) - d^2(v_0, s)) + d(v_0, v_1)(d^2(v_2, s) - d^2(v_0, s)) \geq C > 0$$

holds, where $C = C(v_0, v_1, v_2) = d(v_0, v_2)d(v_0, v_1)(d(v_0, v_1) + d(v_0, v_2))$.

Proof. Denote the left-hand side of (110) by $g(s)$. There are 3 cases to consider.

CASE 1. Assume that s does not belong to the components of v_1 and v_2 . Then

$$\begin{aligned} & d(v_0, v_2)(d^2(v_1, s) - d^2(v_0, s)) + d(v_0, v_1)(d^2(v_2, s) - d^2(v_0, s)) \\ &= d(v_0, v_2)d(v_0, v_1)(2d(v_0, s) + d(v_0, v_1)) + d(v_0, v_1)d(v_0, v_2)(2d(v_0, s) + d(v_0, v_2)) \\ &= d(v_0, v_2)d(v_0, v_1)(4d(v_0, s) + d(v_0, v_1) + d(v_0, v_2)) \\ &\geq d(v_0, v_2)d(v_0, v_1)(d(v_0, v_1) + d(v_0, v_2)) \end{aligned}$$

and hence (110) holds.

CASE 2. Assume that s belongs to the component of v_1 . Define

$$x = x(s) = d(v_1, s) \text{ and } y = y(s) = d(v_0, s).$$

In this notation we get

$$g(s) = g(x, y) = d(v_0, v_2)(x^2 - y^2) + d(v_0, v_1)(2yd(v_0, v_2) + d^2(v_0, v_2)).$$

Dividing by a positive value $d(v_0, v_2)$, we get

$$g(s) > 0 \text{ if and only if } \frac{g(x, y)}{d(v_0, v_2)} = x^2 - y^2 + d(v_0, v_1)(2y + d(v_0, v_2)) > 0.$$

Now, observe that the numbers x , y , and $d(v_0, v_1)$ satisfy triangle inequalities

$$\begin{cases} x + y \geq d(v_0, v_1), \\ x + d(v_0, v_1) \geq y, \\ y + d(v_0, v_1) \geq x, \end{cases}$$

that bound the area visualized in Figure A.3. The function of two variables $\frac{g(x, y)}{d(v_0, v_2)}$ attains the minimal value $d^2(v_0, v_1) + d(v_0, v_1)d(v_0, v_2)$ on the boundary of the specified area. Hence, the inequality $g(s) \geq d(v_0, v_2)d(v_0, v_1)(d(v_0, v_1) + d(v_0, v_2))$ holds for any s in the component of v_1 .

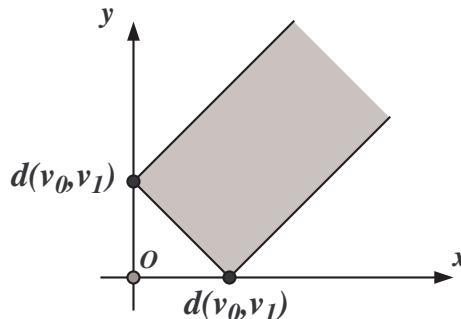


FIGURE A.3. Area of possible triangle side lengths.

CASE 3. If s belongs to the component of v_2 then using same arguments as for the previous case one shows that (110) holds. ■

COROLLARY A.5.2. *Let Γ be a connected graph, v_0 a cut-point in Γ , and v_1, v_2 belong to distinct components of $\Gamma \setminus \{v_0\}$. Then the inequality*

$$d(v_0, v_2)(M(v_1) - M(v_0)) + d(v_0, v_1)(M(v_2) - M(v_0)) \geq C > 0$$

holds, where $C = C(v_0, v_1, v_2) = d(v_0, v_2)d(v_0, v_1)(d(v_0, v_1) + d(v_0, v_2))$.

Proof. Indeed,

$$\begin{aligned} & d(v_0, v_2)(M(v_1) - M(v_0)) + d(v_0, v_1)(M(v_2) - M(v_0)) \\ &= \sum_{s \in V(\Gamma)} (d(v_0, v_2)(d^2(v_1, s) - d^2(v_0, s)) + d(v_0, v_1)(d^2(v_2, s) - d^2(v_0, s)))\mu(s) \\ &\geq \sum_{s \in V(\Gamma)} C\mu(s) = C = d(v_0, v_2)d(v_0, v_1)(d(v_0, v_1) + d(v_0, v_2)), \text{ by Lemma A.5.1} \end{aligned}$$

■

COROLLARY A.5.3 (Cut Point Lemma). *Let Γ be a connected graph, v_0 a cut-point in Γ . If v_1 and v_2 belong to distinct connected components of $\Gamma \setminus \{v_0\}$, then the inequalities $M(v_0) \geq M(v_1)$ and $M(v_0) \geq M(v_2)$ cannot hold simultaneously.*

Proof. Assume to the contrary that $M(v_0) \geq M(v_1)$ and $M(v_0) \geq M(v_2)$ hold simultaneously which is equivalent to $M(v_1) - M(v_0) \leq 0$ and $M(v_2) - M(v_0) \leq 0$. Then, multiplying by positive constants and adding the inequalities above, we get $d(v_0, v_2)(M(v_1) - M(v_0)) + d(v_0, v_1)(M(v_2) - M(v_0)) \leq 0$ which is impossible by Corollary A.5.2. This contradiction finishes the proof. ■

COROLLARY A.5.4 (Mean-set in a graph with a cut-point). *Let v_0 be a cut-point in a graph Γ and $\Gamma \setminus \{v_0\}$ a disjoint union of connected components $\Gamma_1, \dots, \Gamma_k$. Then for any distribution μ on Γ there exists a unique $i = 1, \dots, k$ such that $\mathbb{E}\mu \subseteq V(\Gamma_i) \cup \{v_0\}$.*

COROLLARY A.5.5 (Mean-set in a graph with several cut-points). *Let v_1, \dots, v_n be cut-points in a graph Γ and $\Gamma \setminus \{v_1, \dots, v_n\}$ a disjoint union of connected components $\Gamma_1, \dots, \Gamma_k$. Then for any distribution μ on Γ there exists a unique $i = 1, \dots, k$ such that $\mathbb{E}\mu \subseteq V(\Gamma_i) \cup \{v_1, \dots, v_n\}$.*

COROLLARY A.5.6. *Let G_1 and G_2 be finitely generated groups and $G = G_1 * G_2$ a free product of G_1 and G_2 . Then for any distribution μ on G the set $\mathbb{E}\mu$ is a subset of elements of the forms gG_1 or gG_2 for some element $g \in G$.*

PROPOSITION A.5.7. *Let Γ be a tree and μ a probability measure on $V(\Gamma)$. Then $|\mathbb{E}\mu| \leq 2$. Moreover, if $\mathbb{E}\mu = \{u, v\}$ then u and v are adjacent in Γ .*

Proof. Observe that any points v_1, v_0, v_2 such that v_0 is connected to v_1 and v_2 satisfy the assumptions of the Cut Point Lemma (Corollary A.5.3). Assume that $v_0 \in \mathbb{E}\mu$. At most one of the neighbors of v_0 can belong to $\mathbb{E}\mu$, otherwise we would have 3 connected vertices with equal M values which contradicts the Cut Point Lemma. ■

COROLLARY A.5.8. *Let μ be a probability distribution on a free group F . Then $|\mathbb{E}\mu| \leq 2$.*

In general, the number of central points can be unlimited. To see this, consider the complete graph K_n on n vertices and let μ be a uniform probability distribution on $V(K_n)$. Clearly, $\mathbb{E}\mu = V(K_n)$. Another example of the same type is a cyclic graph C_n on n vertices with a uniform probability distribution μ on $V(C_n)$. Clearly, $\mathbb{E}\mu = V(C_n)$. In all previous examples, the centers in a graph formed a connected subgraph. This is not always the case. One can construct graphs with as many centers as required and property that distances between centers are very large (as large as one wishes).

A.6. Computation of mean-sets in graphs

In this section we discuss computational issues that we face in practice. One of the technical difficulties is that, unlike the average value S_n/n for real-valued random variables, the sample mean-set $\mathbb{S}_n \equiv \mathbb{S}(\xi_1, \dots, \xi_n)$ is hard to compute. Let G be a group and $\{\xi\}_{i=1}^n$ a sequence of random i.i.d. elements taking values in G . Several problems arise when trying to compute \mathbb{S}_n :

- Computation of the set $\{M(g) \mid g \in G\}$ requires $O(|G|^2)$ steps. This is computationally infeasible for large G , and simply impossible for infinite groups. Hence we might want to reduce the search of a minimum to some small part of G .
- There exist infinite groups in which the distance function $|\cdot|$ is very difficult to compute (see more on this issue further in Section A.7.6). The braid group B_∞ is one of such groups. The computation of the distance function for B_∞ is an NP-hard problem, see [221]. Such groups require special treatment. Moreover, there exist infinite groups for which the distance function $|\cdot|$ is not computable. We omit consideration of such groups.

On the other hand, we can try to devise some heuristic procedure for this task. As we show below, if the function M satisfies certain local monotonicity properties, then we can achieve good results. The next algorithm is a simple direct descent heuristic which can be used to compute the minimum of a function f (we restate this algorithm for a given group G and function M_n in Section A.7.7, where we introduce our probabilistic mean-set attack, see Algorithm A.7.1).

ALGORITHM A.6.1 (Direct Descent Heuristic).

INPUT: A graph Γ and a function $f : V(\Gamma) \rightarrow \mathbb{R}$.

OUTPUT: A vertex v that locally minimizes f on Γ .

COMPUTATIONS:

- A. Choose a random $v \in V(\Gamma)$.
- B. If v has no adjacent vertex with smaller value of f , then output current v .
- C. Otherwise put $v \leftarrow u$ where u is any adjacent vertex such that $f(u) < f(v)$. Go to step B.

It turns out that if a function f satisfies certain local properties, then we can achieve good results; namely, the proposed algorithm finds the vertex that minimizes f on Γ exactly. We say that a function $f : V(\Gamma) \rightarrow \mathbb{R}$ is *locally decreasing* if at any vertex $v \in V(\Gamma)$, such that f does not have minimum at v , there exists an adjacent vertex u such that $f(u) < f(v)$. We say that a function f is *locally finite* if for any $a, b \in \mathbb{R}$ the set $f(V(\Gamma)) \cap [a, b]$ is finite.

LEMMA A.6.2. *Let Γ be a graph and $f : V(\Gamma) \rightarrow \mathbb{R}$ a real-valued function that attains its minimum on Γ . If f is locally decreasing and locally finite, then Algorithm A.6.1 for Γ and f finds a vertex that minimizes f on Γ .*

Proof. Let $v \in V(\Gamma)$ be a random vertex chosen by Algorithm A.6.1 at Step A. If v is a minimum of f , then the algorithm stops with the correct answer v . Otherwise, the algorithm, at Step C, chooses any vertex u adjacent to v such that $f(u) < f(v)$. Such a vertex u exists, since the function f is locally decreasing by assumption. Next, Algorithm A.6.1 performs the same steps for u . Essentially, it produces a

succession of vertices v_0, v_1, v_2, \dots such that $v_0 = v$ and, for every $i = 0, 1, 2, \dots$, the vertices v_i, v_{i+1} are adjacent in Γ with the property $f(v_i) > f(v_{i+1})$.

We claim that the constructed succession cannot be infinite. Assume, to the contrary, that the chain v_0, v_1, v_2, \dots is infinite. Let m be the minimal value of f on Γ . Then $f(V(\Gamma)) \cap [m, f(v)]$ is infinite, and, f cannot be locally finite. Contradiction. Hence the sequence is finite, and the last vertex minimizes f on $V(\Gamma)$. ■

LEMMA A.6.3. *Let μ be a distribution on a locally finite graph Γ such that a weight function M is defined. Then the function M is locally finite on Γ .*

Proof. Since the function M is non-negative, it suffices to prove that for any $b \in \mathbb{R}_+$ the set $M(V(\Gamma)) \cap [0, b]$ is finite. Let $v \in \mathbb{E}(\xi)$, i.e., v minimizes the value of M , and $r \in \mathbb{N}$ such that $0 < \frac{1}{2}M(v) \leq \sum_{i \in B_v(r)} d^2(v, i)\mu(i)$, as in the proof of Lemma A.2.4. Choose an arbitrary value $b \in \mathbb{R}_+$ and put $\alpha = \max\{2, b/M(v)\}$. Then one can prove (as in Lemma A.2.4) that for any $u \in \Gamma \setminus B_v((\alpha + 2)r)$, we have $M(u) > (\alpha + 1)M(v) > b$. Therefore, $M(V(\Gamma)) \cap [0, b] \subset M(B_v((\alpha + 2)r))$ and the set $B_v((\alpha + 2)r)$ is finite. ■

THEOREM A.6.4. *Let μ be a distribution on a locally finite tree T such that a function M is totally defined. Then Algorithm A.6.1 for T and M finds a central point (mean-set) of μ on T .*

Proof. Follows from Lemmas A.6.2, A.6.3, A.2.4, and Corollary A.5.3. ■

Note, the function M is not locally decreasing for every graph, and a local minimum, computed by Algorithm A.6.1, is not always a global minimum.

A.6.1. Experiments. In this section we demonstrate how the technique of computing mean-sets, employing the Direct Descent Algorithm A.6.1 described in Section A.6, works in practice and produces results supporting our SLLN for graphs and groups. More precisely, we arrange series of experiments in which we compute the sample mean-sets of randomly generated samples of n random elements and observe a universal phenomenon: the greater the sample size n , the closer the sample mean gets to the actual mean of a given distribution. In particular, we experiment with free groups, in which the length function is easily computable. All experiments are done using the CRAG software package; see [50].

One of the most frequently used distributions on the free groups is a uniform distribution μ_L on a *sphere* of radius L defined as $S_L = \{w \in F(X) \mid |w| = L\}$. Clearly, S_L is finite. Therefore, we can easily define a uniform distribution μ_L on it as

$$\mu_L(w) = \begin{cases} \frac{1}{|S_L|} & \text{if } |w| = L, \\ 0 & \text{otherwise.} \end{cases}$$

The reader interested in the question of defining probabilities on groups can find several approaches to this issue in [28]. One of the properties of μ_L is that its mean-set is just the trivial element of the free group $F(X)$. Observe also that the distance of any element of $F(X)$ to the mean-set is just the length of this element (or length of the corresponding word).

Table 1 below contains the results of experiments for the distributions $\mu_5, \mu_{10}, \mu_{20}, \mu_{50}$ on the group F_4 . The main parameters in our experiments are the rank r of the free group, the length L , and the sample size n . For every particular triple of parameter values (r, L, n) , we perform a series of 1000 experiments to

which we refer (in what follows), somewhat loosely, as series (r, L, n) . Each cell in the tables below corresponds to a certain series of experiments with parameters (r, L, n) . In each experiment from the series (r, L, n) , we randomly generate n words w_1, \dots, w_n , according to distribution μ_L , compute the sample mean-set \mathbb{S}_n , and compute the displacement of the actual center of μ_L from \mathbb{S}_n . The set \mathbb{S}_n is computed using Algorithm A.6.1 which, according to Theorem A.6.4, always produces correct answers for free groups. Every cell in the tables below contains a pair of numbers (d, N) ; it means that in N experiments out of 1000 the displacement from the real mean was d .

One can clearly see from the tables that the bigger the sample size n is, the closer we get to the actual mean value.

$L \setminus n$	2	4	6	8	10	12	14	16
μ_5	(0,885) (1,101) (2,13) (3,1)	(0,943) (1,55) (2,2)	(0,978) (1,22)	(0,988) (1,12)	(0,999) (1,1)	(0,998) (1,2)	(0,1000)	(0,999) (1,1)
μ_{10}	(0,864) (1,117) (2,16) (3,2) (4,1)	(0,930) (1,69) (2,1)	(0,976) (1,24)	(0,993) (1,7)	(0,994) (1,6)	(0,999) (1,1)	(0,1000)	(0,1000)
μ_{20}	(0,859) (1,116) (2,19) (3,6)	(0,940) (1,58) (2,2)	(0,975) (1,25)	(0,985) (1,15)	(0,991) (1,9)	(0,1000)	(0,999) (1,1)	(0,999) (1,1)
μ_{50}	(0,872) (1,108) (2,19) (3,1)	(0,928) (1,71) (2,1)	(0,984) (1,16)	(0,991) (1,9)	(0,998) (1,2)	(0,997) (1,3)	(0,998) (1,2)	(0,999) (1,1)

TABLE 1. The results of experiment for F_4 .

By doing experiments for free groups of higher ranks, one can easily observe that as the rank of the free group grows, we get better and faster convergence. Intuitively, one may think about this outcome as follows: the greater the rank is, the more branching in the corresponding Cayley graph we have, which means that more elements are concentrated in a ball, and the bigger growth (in that sense) causes the better and faster convergence.

At the end, the important conclusion is that these experimental results support the strong law of large numbers for graphs and groups proved in Section A.2.2, and we can say that the law actually works in practice.

A.7. Probabilistic cryptanalysis of Sibert type protocols

The goal of this part of the appendix is to analyze the Sibert et al. group-based (Feige-Fiat-Shamir type) authentication protocol and to show that the protocol is not computationally zero-knowledge. As experimental evidence in Section A.7.8 shows, this approach is practical and can succeed even for groups with no efficiently computable length function such as braid groups (see Section 5.1 for a background on braids as platform groups).

In other words, we explain the practical use of our probabilistic machinery on graphs and groups in detail. We look at the algebraic problem from the probabilistic angle and attack the protocol from a totally unexpected side; it has nothing to do with the originally claimed security assumptions and at no point uses the conjugacy

operation. We obtain the secret information using our strong law of large numbers, the “shift” property, and an algorithm for computing \mathbb{S}_n .

As the reader already knows from the present book, the group-based cryptography attracted a lot of attention after invention of the Anshel-Anshel-Goldfeld [5] and Ko-Lee et al. [161] key-exchange protocols in 1999. Since then a number of new cryptographic protocols, including public-key *authentication protocols*, based on infinite groups were invented and analyzed. Here we consider a particular interactive group-based authentication scheme, Sibert et al. protocol (see Section 8.2, or Appendix Section A.7.7 in this book).

A.7.1. Outline. In Section A.7.2, we briefly recall what zero-knowledge interactive proof systems are about. Next, in Section A.7.3, we talk about slight differences in descriptions of Sibert et al. protocol presented in [256] and [57]. We proceed to describe security assumptions of the protocol in Section A.7.4. Then we explain the idea of our new probabilistic *mean-set attack* and introduce the so-called *search problem* in infinite groups. This task is completed in Section A.7.5. In Section A.7.7 we formulate and prove the main theoretical results of this part of the appendix: mean-set attack principles under different assumptions. The mean-set attack principles give asymptotic bounds on the failure rate in the proposed mean-set attack. We also indicate that even if the proposed algorithm fails, we can still gain some information about the secret key of the Prover. In other words, the more rounds of the protocol are performed, the more information about the secret key we can gain. In Section A.7.8.1, we present results of our experiments with the classical key generation according to [57]. Section A.7.8.2 is concerned with results of experiments with the alternative (special) key generation proposed by Sibert et al. in [256]. At the end, in Section A.7.9, we discuss possible methods for defending against the mean-set attack.

A.7.2. Zero-knowledge interactive proof systems. Recall that *static (non-interactive) proof* is a fixed sequence of statements that are either axioms or are derived from previous statements (see [99]). As opposed to that, any *interactive proof of knowledge system* is a multi-round randomized protocol for two parties, in which one of the parties (the Prover) wishes to convince another party (the Verifier) of the validity of a given assertion. Loosely speaking, we can think of interactive proof as of a game between the Verifier and the Prover consisting of questions asked by the Verifier, to which the Prover must reply “convincingly”. Every interactive proof of knowledge should satisfy *completeness* and *soundness* properties ([74], [100]):

Completeness: If the assertion is true, it should be accepted by the Verifier with high probability.

Soundness: If the assertion is false, then the Verifier rejects it with high probability, i.e., the so-called *soundness error* should be small.

We can say that interactive proofs are *probabilistic proofs* by their nature, because there is some small probability, the soundness error, that a cheating Prover will be able to convince the Verifier of a false statement. However, the soundness error can usually be decreased to negligibly small values by repetitions of steps.

It happens that completeness and soundness properties are not enough to make an interactive proof secure. The Verifier can be a potential adversary (cheating verifier, intruder, eavesdropper) that tries to gain knowledge (secret) from the Prover.

This means that we need another property of the randomized proof (identification protocol, in particular) to ensure that security is preserved. If the Prover does not trust the Verifier and does not want to compromise any private information in the process of providing the proof of identity, then the following property, concerned with the preservation of security, becomes very important:

Zero-Knowledge (ZK): Except the validity of the Prover's assertions, no other information is revealed in the process of the proof.

This property assures the user (the Prover) that he/she does not compromise the privacy of his/her secrets in the process of providing the proof (of identity, for instance). In other words, one can think of *zero-knowledge* as of *preservation of security* on the intuitive level. If a given protocol possesses the zero-knowledge property, then it is considered to be a *zero-knowledge interactive proof system*.

There are three different notions of zero-knowledge that have been commonly used in the literature ([99], [100]); namely, perfect zero-knowledge, statistical zero-knowledge, and computational zero-knowledge. The first notion is the most strict definition of ZK, which is rarely useful in practice. The last notion of the ZK property (computational zero-knowledge) is the most liberal notion, and it is used more frequently in practice than the others.

Sibert et al. authentication protocol, is an example of an interactive (dynamic, randomized) proof system. In what follows below, we use probabilistic tools, introduced in Section A.2, to design an attack on this particular cryptographic primitive and show that it is not computationally zero-knowledge. In addition, we conduct some experiments that support our conclusions and show that the protocol is not secure in practice.

A.7.3. Description of the protocol. The Sibert's protocol is an iterated two-party three-pass Feige-Fiat-Shamir [74] type authentication protocol. There are two slightly different descriptions of the protocol available in [57] and [256] with two different key generation algorithms. In [256], the protocol is introduced as *Scheme II*. Here, we follow the description of the scheme from the survey [57], except for the minor notational modifications in the conjugation. These modifications do not affect the protocol and its cryptographic properties at all (inverting r and y in [57] would resolve it). In addition, [57] and [256] treat the protocol slightly differently themselves, with and without a collision-free one-way hash function, respectively. Nevertheless, it is not essential for our analysis.

Let G be a (non-commutative, infinite) group, called the *platform group* and μ a probability measure on G . The Prover's *private key* is an element $s \in G$, the Prover's *public key* is a pair (w, t) , where w is an arbitrary element of the group G , called the *base element*, and $t = s^{-1}ws$ is a conjugate of w by s . In addition, we assume that H is a collision-free one-way hash function from G to $\{0, 1\}^N$. A single round of the protocol is performed as follows:

- (1) The Prover chooses a random element $r \in G$, called the *nonce*, according to the probability measure μ , and sends $x = H(r^{-1}tr)$, called the *commitment*, to the Verifier.
- (2) The Verifier chooses a random bit c , called the *challenge*, and sends it to the Prover.
 - If $c = 0$, then the Prover sends $y = r$ to the Verifier and the Verifier checks if the equality $x = H(y^{-1}ty)$ is satisfied.

- If $c = 1$, then the Prover sends $y = sr$ to the Verifier and the Verifier checks if the equality $x = H(y^{-1}wy)$ is satisfied.

This round is repeated k times to guarantee the *soundness error* (i.e., probability that a cheating Prover will be able to convince the Verifier of a false statement) of order 2^{-k} , which is considered to be negligible if k is large, say $k \geq 100$. The Sibert's protocol satisfies both, completeness and soundness, properties of interactive proof systems.

In addition, [256] describes another authentication protocol, the so-called *Scheme III*, which is different from the one described above. Even though techniques of the present exposition do not directly apply to that protocol, we believe that using similar ideas, this scheme can be successfully attacked as well.

A.7.4. Security of the protocol. Note that if an intruder (named Eve) can compute the secret element s or any element $s' \in G$ such that $t = s'^{-1}ws'$, i.e., if Eve can solve the *conjugacy search problem* for G , then she can authenticate as the Prover. Thus, as indicated in [256], the computational difficulty of the conjugacy search problem for G is necessary for security of this protocol.

Originally, it was proposed to use braid groups B_n (see Chapter 5, Section 5.1 for a background) as platform groups, because there was no efficient solution of the conjugacy search problem for B_n known. This motivated a lot of research about braid groups. As a result of recent developments ([22], [24], [23]), there is an opinion that the conjugacy search problem for B_n can be solved in polynomial time. If that is true in fact, then the Sibert et al. authentication protocol is insecure for B_n . Nevertheless, the same protocol can be used with other platform groups and, hence, it is important to have tools for analysis of this type of general Sibert protocols. We show below that it is not necessary to solve the conjugacy search problem for G to break the scheme. Instead, one can analyze zero-knowledge property of the protocol by employing ideas from probability theory and show that the protocol is insecure under a mild assumption of existence of an efficiently computable length function for the platform group G . Even for groups with no efficiently computable length function, such as B_n , a reasonable approximation can do the job.

Now, let μ be a probability measure on a platform group G . We say that μ is *left-invariant* if for every $A \subseteq G$ and $g \in G$ the equality $\mu(A) = \mu(gA)$ holds. The following result is proved in [256].

PROPOSITION ([256]). *Let G be a group. If the conjugacy search problem for G is computationally hard (cannot be solved by a probabilistic polynomial time Turing machine) and μ is a left-invariant probability measure on G then the protocol outlined above is a zero knowledge interactive proof system.*

Clearly, there are no left-invariant probability measures on braid groups, used as platform groups in the protocol, and, therefore, as noticed in [57] and [256], this protocol cannot be a perfect zero knowledge interactive proof system when used with an infinite group such as B_n . Nevertheless, it is conjectured in [256] that the scheme can be computationally zero knowledge for certain distributions μ on B_n . The authors supported that conjecture by statistical arguments based on length analysis.

A.7.5. The idea of mean-set attack: the shift search problem. If we look at the protocol outlined in Section A.7.3, we observe that the Prover sends to the Verifier a sequence of random elements of two types: r and sr , where r is a randomly generated element and s is the Prover's secret element. Any passive eavesdropper (Eve) can arrange a table of challenge/response transactions, where each row corresponds to a single round of the protocol, as shown below,

Round	Challenge	Response type # 1	Response type # 2
1	$c = 1$	—	sr_1
2	$c = 0$	r_2	—
3	$c = 0$	r_3	—
4	$c = 1$	—	sr_4
5	$c = 0$	r_5	—
...
n	$c = 0$	r_n	—

and obtain two sets of elements, corresponding to $c = 0$ and $c = 1$, respectively: $R_0 = \{r_{i_1}, \dots, r_{i_k}\}$ and $R_1 = \{sr_{j_1}, \dots, sr_{j_{n-k}}\}$, where all elements r_i are distributed according to μ , i.e., all these elements are generated by the same random generator. Eve's goal is to recover the secret element s based on the intercepted sequences R_0 and R_1 . We call this problem a *shift search problem*.

To explain the idea of the *mean-set attack*, assume for a moment that the group G is an infinite cyclic group \mathbb{Z} . In that case, we can rewrite the elements of R_1 in additive notation $\{s + r_{j_1}, \dots, s + r_{j_{n-k}}\}$. Then we can compute the empirical average $\bar{r}_0 = \frac{1}{k} \sum_{m=1}^k r_{i_m}$ of the elements in $R_0 \subset \mathbb{Z}$ and the empirical average $\bar{r}_1 = \frac{1}{n-k} \sum_{l=1}^{n-k} (s + r_{j_l}) = s + \frac{1}{n-k} \sum_{l=1}^{n-k} r_{j_l}$ of the elements in $R_1 \subset \mathbb{Z}$. By the strong law of large numbers for real-valued random variables the larger the sequence R_0 is, the closer the value of \bar{r}_0 to the actual mean $\mathbb{E}(\mu)$ of the distribution μ on \mathbb{Z} , induced by r . Similarly, the larger the sequence R_1 is, the closer the value of \bar{r}_1 is to the number $s + \mathbb{E}(\mu)$. Therefore, subtracting \bar{r}_0 from \bar{r}_1 , we obtain a good guess of what s is. Observe three crucial properties that allow us to compute the secret element in the case $G = \mathbb{Z}$:

- (AV1)** (Strong law of large numbers for real-valued random variables) If $\{\xi_i\}_{i=1}^\infty$ is a sequence of independent and identically distributed (i.i.d.) real-valued random variables, then

$$\frac{1}{n} \sum_{i=1}^n \xi_i \xrightarrow{\text{P}} \mathbb{E}\xi_1$$

with probability one as $n \rightarrow \infty$, provided $\mathbb{E}(\xi_1) < \infty$.

- (AV2)** (“Shift” property or linearity) For any real-valued random variable ξ , the formula

$$\mathbb{E}(c + \xi) = c + \mathbb{E}(\xi)$$

holds.

- (AV3)** (Efficient computations) The average value $\frac{1}{n} \sum_{i=1}^n \xi_i$ is efficiently computable.

Geometrically, we can interpret this approach as follows. Given a large sample of random, independent, and identically distributed points r_{i_1}, \dots, r_{i_k} and a large

sample of shifted points $s+r_{j_1}, \dots, s+r_{j_{n-k}}$ on the real line, the shift s is “effectively visible”.

As we have seen in Section A.2, the same is true in general infinite groups and we can generalize a number of mathematical tools of the classical probability theory to finitely generated groups in order to have the counterparts of (AV1), (AV2), and (AV3). Recall that for a random group element $\xi : \Omega \rightarrow G$ (see Section A.2.2), one can define a set $\mathbb{E}(\xi) \subseteq G$ called the *mean-set* (average, expectation). Then, for a sample of n random group elements ξ_1, \dots, ξ_n , one can define their average — a set $\mathbb{S}_n = \mathbb{S}(\xi_1, \dots, \xi_n) \subseteq G$ called the *sample mean-set* of elements ξ_1, \dots, ξ_n , so that we have a “shift” property $\mathbb{E}(s\xi) = s\mathbb{E}(\xi)$ and a generalization of the strong law of large numbers (SLLN) for groups with respect to $\mathbb{E}(\xi)$ in a sense that $\mathbb{S}(\xi_1, \dots, \xi_n)$ converges to $\mathbb{E}(\xi_1)$ as $n \rightarrow \infty$ with probability one (see Section A.3 for precise definitions and statements). In addition, assume that sample mean $\mathbb{S}(\xi_1, \dots, \xi_n)$ is efficiently computable.

Using the operator \mathbb{S} , Eve can compute a set

$$\mathbb{S}(sr_{j_1}, \dots, sr_{j_{n-k}}) \cdot [\mathbb{S}(r_{i_1}, \dots, r_{i_k})]^{-1},$$

which should contain s with high probability when n is sufficiently large. This is the idea of the mean-set attack and our approach to the *shift search problem*. Furthermore, one can show that the more rounds of the protocol are performed, the more information about the secret key our attack gains (note that at the same time the protocol is iterated by its nature, and large number of rounds is important for its reliability in a sense of the soundness property). The discussion above leads to the so-called *mean-set attack principle*, which is formulated and proved for different assumptions below, in Section A.7.7.

A.7.6. Effective computation of a mean-set. Let G be a group and $\{\xi_i\}_{i=1}^n$ a sequence of random i.i.d. elements taking values in G such that the corresponding weight function $M(\cdot)$ is totally defined. Recall from Section A.6 that one of the technical difficulties encountered in practice is that, unlike the classical average value $(x_1 + \dots + x_n)/n$ for real-valued random variables, the sample mean-set \mathbb{S}_n is hard to compute. In other words, in general, our definition of the mean-set (Definition A.2.1) might not satisfy the property (AV3). The problems that arise when trying to compute \mathbb{S}_n were discussed in Section A.6. In the same section we devised a heuristic procedure Direct Descent Algorithm A.6.1 to solve one of the problems. As proved in A.6, if the weight function $M(\cdot)$ (introduced in Section A.2.2) satisfies certain local monotonicity properties, then our procedure achieves the desired result. For the convenience of the reader, we restate our simple direct descent heuristic algorithm here. This time we use the sample weight function M_n (introduced in Section A.2.2) that comes from a sample of random group elements $\{g_1, \dots, g_n\}$ for a finitely-generated group G .

ALGORITHM A.7.1 (Direct Descent Heuristic for G).

INPUT: A group G with a finite set of generators $X \subseteq G$ and a sequence of elements $\{g_1, \dots, g_n\}$ in G .

OUTPUT: An element $g \in G$ that locally minimizes $M_n(\cdot)$.

COMPUTATIONS:

- A. Choose a random $g \in G$ according to some probability measure ν on G .
- B. If for every $x \in X^{\pm 1}$, $M_n(g) \leq M_n(gx)$, then output g .

- C. Otherwise put $g \leftarrow gx$, where $x \in X^{\pm 1}$ is an element minimizing the value of $M_n(gx)$ and go to step B.

As any other direct descend heuristic method, Algorithm A.7.1 might not work if the function M_n has local minima. It is proved in A.6 (A.6.4) that it always works for trees and, hence, for free groups.

The second problem of computing \mathbb{S}_n (discussed in Section A.6) concerns practical computations of length function in G . It turns out that we need a relatively mild assumption to deal with it — the existence of an efficiently computable distance function $d_X(\cdot, \cdot)$; even a “reasonable” approximation of the length function may work. In this work we approximate geodesic length using the method described in [192]. Even though it does not guarantee the optimal result, it was proved to be practically useful in a series of attacks; see [193, 212, 211, 175].

A.7.7. The mean-set attack. In this section, we use theoretical results stated above to attack the Sibert et al. protocol, described in Section A.7.3. In the following heuristic attack we use Algorithm A.7.1 to compute sample mean-set \mathbb{S}_n .

ALGORITHM A.7.2 (The mean-set attack).

INPUT: The Prover’s public element (t, w) and sequences R_0 and R_1 as in the protocol.

OUTPUT: An element z satisfying the equality $t = z^{-1}wz$ (which can be considered as the Prover’s private key), or *Failure*.

COMPUTATIONS:

- A. Apply Algorithm A.7.1 to R_0 and obtain g_0 .
- B. Apply Algorithm A.7.1 to R_1 and obtain g_1 .
- C. If $g_1 g_0^{-1}$ satisfies $t = (g_1 g_0^{-1})^{-1}w(g_1 g_0^{-1})$ then output $g_1 g_0^{-1}$. Otherwise output *Failure*.

If the algorithm outputs an element $z \in G$, then z can serve as the Prover’s original secret s ; any solution of the conjugacy equation $t = x^{-1}wx$ does. In general, z can be different from s , and there are no means for the adversary to determine whether $z = s$. In spite of that, Eve, who is only trying to authenticate as the Prover, considers this z a success. On the other hand, since our goal is to show that the protocol is not computationally zero-knowledge, we estimate the probability to find s . Only this original secret element s is considered as a success in our analysis. Other outcomes that work for Eve (when $z \neq s$) are ignored.

The theorems below give asymptotic bounds on the failure rate (for the original s) in the mean-set attack. We show that the probability of the failure can decrease linearly or exponentially, depending on the distribution μ .

THEOREM A.7.3 (Mean-set attack principle – I). *Let G be a group, X a finite generating set for G , $s \in G$ a secret fixed element, and ξ_1, ξ_2, \dots a sequence of randomly generated i.i.d. group elements, such that $\mathbb{E}(\xi_1) = \{g\}$. If ξ_1, \dots, ξ_n is a sample of random elements of G generated by the Prover, c_1, \dots, c_n a succession of random bits (challenges) generated by the Verifier, and*

$$y_i = \begin{cases} r_i & \text{if } c_i = 0, \\ sr_i & \text{if } c_i = 1 \end{cases}$$

random elements representing responses of the Prover, then there exists a constant D such that

$$\mathbf{P}\left(s \notin \mathbb{S}(\{y_i \mid c_i = 1, i = 1, \dots, n\}) \cdot \mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\})^{-1}\right) \leq \frac{D}{n}.$$

Proof. It follows from Theorem A.4.2 that there exists a constant C such that

$$\mathbf{P}(\mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\}) \neq \{g\}) \leq \frac{C}{|\{i \mid c_i = 0, i = 1, \dots, n\}|}.$$

Applying Chebyshev's inequality to Bernoulli random variables $\{c_i\}$ having $\mathbb{E}(c_i) = \frac{1}{2}$ and $\sigma_{c_i}^2 = \frac{1}{4}$, we obtain

$$\mathbf{P}\left(|\{i \mid c_i = 0, i = 1, \dots, n\}| < \frac{n}{4}\right) < \frac{4}{n}.$$

In more detail, if the number of zeros in our sample of challenges is less than $\frac{n}{4}$, then the number of ones is greater or equal to $\frac{3n}{4}$, and we have

$$\mathbf{P}\left(\left|\{i \mid c_i = 0, i = 1, \dots, n\}\right| < \frac{n}{4}\right) < \mathbf{P}\left(\left|\sum_{i=1}^n c_i - \frac{n}{2}\right| \geq \frac{n}{4}\right).$$

Note that

$$\left|\sum_{i=1}^n c_i - \frac{n}{2}\right| \geq \frac{n}{4} \Leftrightarrow \left|\frac{\sum_{i=1}^n c_i}{n} - \frac{1}{2}\right| \geq \frac{1}{4}$$

and

$$\mathbf{P}\left(\left|\frac{\sum_{i=1}^n c_i}{n} - \frac{1}{2}\right| \geq \frac{1}{4}\right) \leq \frac{4}{n}$$

from the classical Chebyshev inequality for sample means with $\varepsilon = \frac{1}{4}$.

It follows that

$$\mathbf{P}(\mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\}) \neq \{g\}) \leq \frac{4}{n} + \frac{4C}{n} \leq \frac{4+4C}{n}.$$

Similarly, we prove that $\mathbf{P}(\mathbb{S}(\{y_i \mid c_i = 1, i = 1, \dots, n\}) \neq \{sg\}) \leq \frac{4+4C}{n}$. Hence,

$$\mathbf{P}(s \notin \mathbb{S}(\{y_i \mid c_i = 1, i = 1, \dots, n\}) \cdot \mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\})^{-1}) \leq \frac{8+8C}{n}. \quad \blacksquare$$

Furthermore, we can get Chernoff-like asymptotic bound if we impose one restriction on distribution μ . Recall the original Hoeffding's inequality ([132]) well known in probability theory. Assume that $\{x_i\}$ is a sequence of independent random variables and that every x_i is almost surely bounded, i.e., $\mathbf{P}(x_i - \mathbb{E}x_i \in [a_i, b_i]) = 1$ for some $a_i, b_i \in \mathbb{R}$. Then for the sum $S_n = x_1 + \dots + x_n$, the inequality

$$\mathbf{P}(S_n - \mathbb{E}S_n \geq n\varepsilon) \leq \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

holds. If x_i are identically distributed, then we get the inequality

$$(111) \quad \mathbf{P}\left(\frac{1}{n}(x_1 + \dots + x_n) - \mathbb{E}x_1 \geq \varepsilon\right) \leq 2 \exp\left(-\frac{2\varepsilon^2}{(b-a)^2}n\right).$$

Now we can prove the mean-set attack principle with exponential bounds.

THEOREM A.7.4 (Mean-set attack principle – II). *Let G be a group, X a finite generating set for G , $s \in G$ a secret fixed element, and ξ_1, ξ_2, \dots a sequence of randomly generated i.i.d. group elements, such that $\mathbb{E}(\xi_1) = \{g\}$. If ξ_1, \dots, ξ_n is a sample of random elements of G generated by the Prover, c_1, \dots, c_n a succession of random bits (challenges) generated by the Verifier,*

$$y_i = \begin{cases} r_i & \text{if } c_i = 0, \\ sr_i & \text{if } c_i = 1 \end{cases}$$

random elements representing responses of the Prover, and the distribution μ has finite support, then there exists a constant $D = D(G, \mu)$ such that

$$\mathbf{P}\left(s \notin \mathbb{S}\left(\{y_i \mid c_i = 1, i = 1, \dots, n\}\right) \cdot \mathbb{S}\left(\{y_i \mid c_i = 0, i = 1, \dots, n\}\right)^{-1}\right) \leq O(e^{-Dn}).$$

Proof. It follows from Theorem A.4.4 that there exists a constant C such that

$$\mathbf{P}(\mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\}) \neq \{g\}) \leq O(e^{-C|\{i \mid c_i = 0, i = 1, \dots, n\}|}).$$

Applying inequality (111) to Bernoulli random variables $\{c_i\}$, we get

$$\mathbf{P}\left(\sum_{i=1}^n c_i - \frac{1}{2} > \frac{1}{4}\right) < e^{-n/8}.$$

Thus, we obtain a bound

$$\mathbf{P}(\mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\}) \neq \{g\}) \leq e^{-n/8} + O(e^{-Cn/4}).$$

Similarly, we prove that $\mathbf{P}(\mathbb{S}(\{y_i \mid c_i = 1, i = 1, \dots, n\}) \neq \{sg\}) \leq e^{-n/8} + O(e^{-Cn/4})$. Hence,

$$\mathbf{P}(s \notin \mathbb{S}(\{y_i \mid c_i = 1, i = 1, \dots, n\}) \cdot \mathbb{S}(\{y_i \mid c_i = 0, i = 1, \dots, n\})^{-1}) \leq O(e^{-Dn})$$

where $D = \min\{1/8, C/4\}$. ■

Algorithm A.7.2 can fail. Nevertheless the pair of the obtained elements g_0, g_1 often encodes some additional information about the secret s . Indeed, assume that $\mathbb{E}\mu = \{g\}$. The element g_0 obtained at step A of Algorithm A.7.2 can be viewed as a product ge_0 for some $e_0 \in G$. Similarly, the element g_1 can be viewed as a product sge_1 for some $e_1 \in G$. Hence Algorithm A.7.2 outputs the secret element s whenever $g_1g_0^{-1} = sge_1e_0^{-1}g^{-1} = s$, i.e., whenever $e_1e_0^{-1} = 1$.

Now, assume that Algorithm A.7.2 has failed, i.e., $e_1e_0^{-1} \neq 1$. In this case, one can try to reconstruct the secret element s as a product

$$g_1 \cdot e \cdot g_0^{-1} = sge_1 \cdot e \cdot e_0^{-1}g^{-1}$$

where e is an unknown element of the platform group. Clearly, e gives a correct answer if and only if $e_1 \cdot e \cdot e_0^{-1} = 1$ or $e = e_1^{-1}e_0$. The element

$$(112) \quad e_1^{-1}e_0$$

is called *the error of the method*. Clearly, one only needs to enumerate all words e of length up to $|e_1^{-1}e_0|$ to reconstruct the required s in the form $g_1eg_0^{-1}$. If a secret element s is chosen uniformly as a word of length l and $|e_1^{-1}e_0| < l$, then we gain some information about s , since the search space for s reduces. We can improve Algorithm A.7.2 by adding such enumeration step as follows.

ALGORITHM A.7.5 (The attack–2).

INPUT: The Prover’s public element (t, w) . Sequences R_0 and R_1 as in the protocol. The number $k \in \mathbb{N}$ – the expected length of error element $e_1 e_0^{-1}$.

OUTPUT: An element z satisfying the equality $t = z^{-1}wz$ (which can be considered as the Prover’s private key), or *Failure*.

COMPUTATIONS:

- A. Apply Algorithm A.6.1 to R_0 and obtain g_0 .
- B. Apply Algorithm A.6.1 to R_1 and obtain g_1 .
- C. For every word e of lengths up to k , check if $g_1 e g_0^{-1}$ satisfies the equality $t = (g_1 e g_0^{-1})^{-1} w (g_1 e g_0^{-1})$ and if so output $g_1 e g_0^{-1}$. Otherwise output *Failure*.

A.7.8. Experiments. To demonstrate the practical use of our mean-set attack, we perform a series of experiments, which we describe below. In [256], [57] two different methods of generation of nonce elements were proposed, both with the same platform group B_n , which has the following (Artin’s) presentation (see Section 5.1)

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{ll} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j & \text{if } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i & \text{if } |i - j| > 1 \end{array} \right\rangle.$$

We distinguish between the two ways, classical ([57]) and alternative ([256]), to generate elements of the underlying group by performing two different sets of experiments outlined below in Sections A.7.8.1 and A.7.8.2. In both cases, we observe that the secret information of the Prover is not secure, and the probability to break the protocol grows as the number of rounds of the protocol increases. All experiments are done using the CRAG software package [50].

A.7.8.1. Classical key generation. Classical key generation of the elements of B_n was suggested in [57] with parameters $n = 50$ (rank of the braid group) and the lengths of private keys $L = 512$. The length function relative to the Artin generators $\{\sigma_1, \dots, \sigma_{n-1}\}$ is *NP-hard*. That is why in this Appendix, as it was already mentioned in Section A.7.6, we use the approximation of geodesic length method, proposed in [193]. See [193, 212, 211, 175] for a series of successful attacks using this method. We want to emphasize that we compute the sampling weight values in the Algorithm A.7.1, which is a subroutine in Algorithm A.7.2, using the approximated distance function values in B_n .

One of the disadvantages of the approximation algorithm that we used is that there is no polynomial time upper bound for that as it uses Dehornoy handle-free forms (see Section 5.1 in the present book and [55]). As a result we do not know the complexity of our algorithm and we do not know how our algorithm scales with parameter values. In each experiment we randomly generate an instance of the authentication protocol and try to break it, i.e., find the private key, using the techniques developed in previous sections. Recall that each authentication is a series of k 3-pass commitment-challenge-response rounds. Therefore, an instance of authentication consists of k triples (x_i, c_i, r_i) , $i = 1, \dots, k$ obtained as described in Section A.7.3. Here x_i is a commitment, c_i is a challenge, and r_i is a response. A random bit c_i is chosen randomly and uniformly from the set $\{0, 1\}$. In our experiments we make an assumption that exactly half of c_i ’s are 0 and half are 1. This allows us to see an instance of the protocol as a pair of equinumerous sets $R_0 = \{r_1, \dots, r_{k/2}\} \subset B_n$ and $R_1 = \{sr'_1, \dots, sr'_{k/2}\} \subset B_n$.

The main parameters for the system are the rank n of the braid group, the number of rounds k in the protocol, and the length L of secret keys. We generate a single instance of the problem with parameters (n, k, L) as follows:

- A braid s is chosen randomly and uniformly as a word of length L over a group alphabet $\{\sigma_1, \dots, \sigma_{n-1}\}$. This braid is a secret element which is used only to generate further data and to compare the final element to.
- A sequence $R_0 = \{r_1, \dots, r_{k/2}\}$ of braid words chosen randomly and uniformly as words of length L over a group alphabet $\{\sigma_1, \dots, \sigma_{n-1}\}$.
- A sequence $R_1 = \{sr'_1, \dots, sr'_{k/2}\}$ of braid words, where r'_i are chosen randomly and uniformly as words of length L over a group alphabet $\{\sigma_1, \dots, \sigma_{n-1}\}$.

For every parameter set (n, k, L) we generate 1000 random instances (R_0, R_1) and run Algorithm A.7.2 which attempts to find the secret key s used in the generation of R_1 .

Below we present the results of actual experiments done for groups B_5 , B_{10} , and B_{20} . Horizontally we have increasing number of rounds k from 10 to 320 and vertically we have increasing lengths L from 10 to 100. Every cell contains a pair $(P\%, E)$ where P is a success rate and E is an average length of the error (112) of the method for the corresponding pair (L, k) of parameter values. All experiments were performed using CRAIG library [50]. The library provides an environment to test cryptographic protocols constructed from non-commutative groups, for example, the braid group.

L \ k	10	20	40	80	160	320
10	(19%, 1.3)	(72%, 0.3)	(97%, 0.04)	(100%, 0)	(100%, 0)	(100%, 0)
50	(2%, 13.4)	(8%, 9)	(68%, 1.3)	(93%, 0.1)	(100%, 0)	(100%, 0)
100	(0%, 53.7)	(0%, 48.1)	(6%, 26.9)	(44%, 14)	(65%, 14.7)	(87%, 5)

TABLE 2. Experiments in B_5 .

L \ k	10	20	40	80	160	320
10	(15%, 1.8)	(68%, 0.3)	(98%, 0)	(100%, 0)	(100%, 0)	(100%, 0)
50	(0%, 4.5)	(23%, 1.3)	(82%, 0)	(97%, 0)	(99%, 0)	(100%, 0)
100	(1%, 41)	(7%, 23.5)	(33%, 5)	(79%, 1)	(97%, 0.6)	(98%, 1.1)

TABLE 3. Experiments in B_{10} .

L \ k	10	20	40	80	160	320
10	(15%, 1.6)	(87%, 0.1)	(100%, 0)	(100%, 0)	(100%, 0)	(100%, 0)
50	(0%, 5.4)	(23%, 1.7)	(81%, 0.2)	(100%, 0)	(100%, 0)	(100%, 0)
100	(0%, 7.8)	(15%, 2)	(72%, 0.3)	(97%, 0)	(100%, 0)	(100%, 0)

TABLE 4. Experiments in B_{20} .

We immediately observe from the data above that:

- the success rate increases as the number of rounds (sample size) increases;
- the success rate decreases as the length of the key increases;
- the success rate increases as the rank of the group increases;
- the average error length decreases as we increase the number of rounds.

The first observation is the most interesting since the number of rounds is one of the main reliability parameters of the protocol, namely, the soundness error decreases as $1/2^k$ as the number of rounds k gets larger. But, at the same time, we observe that security of the scheme decreases as k increases. The second observation can be interpreted as follows; the longer the braids are the more difficult it is to compute the approximation. The third observation is easy to explain. The bigger the rank of the group the more braid generators commute and the simpler random braids are.

A.7.8.2. *Alternative key generation.* As we have mentioned in Section A.7.4, the Sibert et al. scheme, proposed in [256], does not possess perfect zero knowledge property. Nevertheless, the authors of [256] try to achieve computational zero knowledge by proposing a special way of generating public and private information. They provide some statistical evidence that the scheme can be computationally zero knowledge if this alternative key generation is used. In this section we, first, outline the proposed key generation method and, second, present actual experiments supporting our theoretical results even for this special key generation method.

The method of generating of braids in [256] can be translated to the notation of the present work as follows. The Prover generates

- nonce elements r as products of L uniformly chosen permutation braids p_i (see [69]) from B_n ,

$$r = p_1 \dots p_L,$$

in particular, r belongs to the corresponding positive monoid.

- the secret key s as the inverse of a product of L uniformly chosen *permutation braids* from B_n , i.e.,

$$s = p_1^{-1} \dots p_L^{-1}.$$

A very useful observation was made when doing the experiments with so generated nonce elements r . We observed that the mean-set in this case is often a singleton set of the form $\{\Delta^k\}$, where Δ is a half-twist braid and $k \in \mathbb{N}$. Therefore, to enhance the performance of Algorithm A.6.1 in step B, we test not only generators $x \in X^{\pm 1}$, but also $x = \Delta$, and if (in step C) Δ minimizes the value of $M_n(gx)$, then we put $x \rightarrow x\Delta$ and return to step B.

In fact, it is an interesting question if the uniform distribution on a sphere in a Garside monoid G^+ has a singleton mean set $\{\Delta_{G^+}^k\}$ for some $k \in \mathbb{N}$, where Δ_{G^+} is the Garside element, Δ , in G^+ ? This is clearly true for free abelian monoids. As we mention above, experiments show that the same can be true in the braid monoid.

Below we present the results of actual experiments done for the group B_{10} . Horizontally we have increasing number of rounds k from 10 to 320 and vertically we have increasing lengths L (in permutation braids) from 3 to 10. Every cell contains a pair $(P\%, E)$ where P is a success rate and E is the average length of the error for the corresponding pair (L, k) of parameter values.

Since the average Artin length (denoted L' in the tables below) of a permutation braid on n strands is of order n^2 , the length of nonce elements grows very fast with

L ; it is shown in the leftmost column of the tables in parentheses. For instance, we can see that for B_{10} the average length of a product of $L = 3$ permutation braids is 81, the average length of a product of $L = 5$ permutation braids is 138, etc.

$L(L') \setminus k$	10	20	40	80	160	320
3 (81)	(0%, 24.6)	(0%, 22.5)	(1%, 19.6)	(4%, 16)	(7%, 13.1)	(25%, 12.3)
5 (138)	(0%, 46.7)	(0%, 40.9)	(0%, 32.5)	(2%, 23.3)	(10%, 17.6)	(28%, 14.2)
10 (274)	(0%, 110.2)	(0%, 102.6)	(0%, 103.5)	(0%, 96.3)	(0%, 92.7)	(0%, 87.9)

TABLE 5. Success rate and average length of the error for experiments in B_{10} .

Again, we observe that success rate increases as we increase the number of rounds, and the average error length decreases as we increase the number of rounds.

A.7.9. Defending against the attack. In this section, we describe several principles one can follow in order to defend against the mean-set attack presented in this appendix or, at least, to make it computationally infeasible. Defending can be done through a special choice of the platform group G or a special choice of a distribution μ on G . Another purpose of this section is to motivate further study of distributions on groups and computational properties of groups.

A.7.9.1. Groups with no efficiently computable length functions. One of the main tools in our technique is an efficiently computable function $d_X(\cdot, \cdot)$ on G . To prevent the attacker from computing mean-sets, one can use a platform group G with a hardly computable length function $d_X(\cdot, \cdot)$ relative to any “reasonable” finite generating set X . By reasonable generating set we mean a set, which is small relative to the main security parameter. Examples of such groups exist. For instance, length function for any finitely presented group with unsolvable word problem is not computable. On the other hand, it is hard to work with such groups, as they do not have efficiently computable normal forms.

A more interesting example is a multiplicative group of a prime field \mathbb{Z}_p^* . The group \mathbb{Z}_p^* is cyclic, i.e., $\mathbb{Z}_p^* = \langle a \rangle$ for some primitive root a of p . It is easy to see that the length of an element $b \in \mathbb{Z}_p^*$ satisfies

$$|b| = \begin{cases} \log_a b & \text{if } \log_a b \leq (p-1)/2, \\ p-1 - \log_a b & \text{otherwise,} \end{cases}$$

and hence the problem of computing the length of an element and the discrete logarithm problem are computationally equivalent. The discrete logarithm problem is widely believed to be computationally hard and is used as a basis of security of many cryptographic protocols, most notably the ElGamal (Section 1.3 in this book or [68]) and the Cramer-Shoup [47] cryptosystems. In other words, \mathbb{Z}_p^* is another example of a group with hardly computable length function.

A.7.9.2. Systems of probability measures. Let G be a platform group. Recall that our assumption was that the Prover uses a fixed distribution on the set of nonce elements, i.e., every element r_i is generated using the same random generator. Instead he can use a sequence of probability measures $\{\mu_i\}_{i=1}^\infty$, where each measure μ_i , $i = 1, 2, \dots$, is not used more than once (ever), i.e., every nonce r_i , $i = 1, 2, \dots$, is generated using a unique distribution $\{\mu_i\}$. In this case, the attacker does not have theoretical grounds for working with sampling mean-sets. Nevertheless, it

can turn out that the sequence of random elements r_1, r_2, \dots can have some other distribution μ^* and the attack will work. Another difficulty with implementing this idea is that there is no systematic study of distributions on general finitely generated groups and, in particular, braid groups. So, it is hard to propose some particular sequence of probability distributions. Some aspects of defining probability measures on infinite groups are discussed in [28] and [29].

A.7.9.3. *Undefined mean-set.* Another way to foil the attack is to use a distribution μ on G such that $\mathbb{E}(\mu)$ is not defined, i.e., the corresponding weight function is not totally defined. In that case the assumption of Theorem A.7.3 fails, and it is easy to see that the sampling weights $M_n(g)$ tend to ∞ with probability 1. Nevertheless, we still can compare the sampling weight values, as explained in [204] and [205], where it is shown that the condition of finiteness of $M^{(2)}$ can be relaxed to that of finiteness of $M^{(1)}$. If $M^{(1)}$ is not defined then that means that the lengths of commitments are too large and are impractical.

A.7.9.4. *Large mean-set.* Also, to foil the attack one can use a distribution μ on G such that the set $\mathbb{E}\mu$ is large. As an example consider an authentication protocol in [243], based on the difficulty of computing discrete discrete logarithms in groups of prime order. The space of nonce elements in [243] is an additive group \mathbb{Z}_q acting by exponentiations on a bigger group \mathbb{Z}_p^* . It is easy to compute length in $(\mathbb{Z}_q, +) = \langle 1 \rangle$. But, since the nonce elements $r \in \mathbb{Z}_q$ are chosen uniformly, it follows that the mean-set is the whole group \mathbb{Z}_q (the uniform measure is right-invariant) and in this case it is impossible to detect the shift s and the mean-set attack fails. We also refer to [229] for a modification of [243] where nonce elements are not taken modulo q and security proof requires a boundary on the number of times the same key is used.

Now, let G be an infinite group. It is impossible to generate elements of G uniformly, but one can try to achieve the property described below that can foil the mean-set attack. Choose a probability measure μ on G so that the mean-set set $\mathbb{E}\mu$ is large. Recall that Algorithm A.7.2 can find up to one element of G minimizing the weight function. For that it uses Algorithm A.7.1 which randomly (according to some measure ν) chooses an element of $g \in G$ and then gradually changes it (descends) to minimize its M value. This way the distribution ν on the initial choices $g \in G$ defines a distribution ν_μ^* on the set of local minima of M on G . More precisely, for $g' \in G$,

$$\nu_\mu^*(g') = \mu\{g \in G \mid \text{Algorithm A.7.1 stops with the answer } g' \text{ on input } g\}.$$

Denote by μ_s the *shifted probability measure* on G by an element s defined by $\mu_s(g) = \mu(s^{-1}g)$. If $S \subseteq G$ is the set of local minima of the weight function M relative to μ then the set sS is the set of local minima relative to μ_s . But the distribution $\nu_{\mu_s}^*$ does not have to be induced from ν_μ^* by the shift s , i.e., the equality $\nu_{\mu_s}^*(g) = \nu_\mu^*(s^{-1}g)$ does not have to hold. In fact, the distributions ν_μ^* and $\nu_{\mu_s}^*$ can “favor” unrelated subsets of S and sS , respectively. That would definitely foil the mean-set attack presented above. On the other hand, if ν_μ^* and $\nu_{\mu_s}^*$ are related, then the mean-set attack can still work.

Finally, we want to mention again that probability measures on groups were not extensively studied and there are no good probability measures known on general groups and no general methods to construct measures satisfying the desired properties. Moreover, the problem of making distributions with large mean-sets is

very complicated because not every subset of a group G can be realized as a mean-set. See Section A.5 for more details. A number of open questions arise regarding the problems mentioned above, but dealing with them is beyond the scope of this Appendix.

Bibliography

- [1] S. I. Adyan and V. G. Durnev, *Decision problems for groups and semigroups*, Russ. Math. Surv. 55 (2000), pp. 207–296.
- [2] S. Ahmad, *Eléments aléatoires dans les espaces vectoriels topologiques*, Ann. Inst. Henri Poincaré 2 (1965), pp. 95–135.
- [3] J. Almeida, *Semidirect products of pseudovarieties from the universal algebraist's point of view*, J. Pure Appl. Algebra 60 (1989), pp. 113–128.
- [4] J. Almeida and P. Weil, *Free profinite semigroups over semidirect products*, Izvestiya VUZ Matematika 39 (1995), pp. 3–31.
- [5] I. Anshel, M. Anshel, and D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Res. Lett. 6 (1999), pp. 287–291.
- [6] I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux, *Key Agreement, The Algebraic EraserTM, and Lightweight Cryptography*. Algebraic Methods in Cryptography, Contemporary Mathematics 418, pp. 1–34. American Mathematical Society, 2006.
- [7] K. Appel and P. Schupp, *Artin groups and infinite Coxeter groups*, Invent. Math. 72 (1983), pp. 201–220.
- [8] G. Arzhantseva and A. Olshanskii, *Genericity of the class of groups in which subgroups with a lesser number of generators are free (Russian)*, Mat. Zametki 59 (1996), pp. 489–496.
- [9] K. Auinger and B. Steinberg, *The geometry of profinite graphs with applications to free groups and finite monoids*, Trans. Amer. Math. Soc. 356 (2004), pp. 805–851.
- [10] ———, *A constructive version of the Ribes-Zaleskii product theorem*, Math. Z. 250 (2005), pp. 287–297.
- [11] ———, *Constructing divisions into power groups*, Theoret. Comput. Sci. 341 (2005), pp. 1–21.
- [12] S. Bachmuth and H. Y. Mochizuki, *Aut(F) \rightarrow Aut(F/F'') is surjective for free group F of rank ≥ 4* , Trans. Amer. Math. Soc. 262 (1985), pp. 81–101.
- [13] G. Baumslag, A. G. Myasnikov, and V. Shpilrain, *Open problems in combinatorial group theory. Second Edition*. Combinatorial and geometric group theory, Contemporary Mathematics 296, pp. 1–38. American Mathematical Society, 2002.
- [14] A. Beck, *On the strong law of large numbers*. Ergodic Theory, Proceedings of the International Symposium, pp. 21–53. Academic Press, New York, 1963.
- [15] A. Beck and D. Giesy, *P-uniform convergence and a vector-valued strong law of large numbers*, Trans. Amer. Math. Soc. 147 (1970), pp. 541–559.
- [16] S. Ben David, B. Chor, O. Goldreich, and M. Luby, *On the theory of average case complexity*, J. Comput. Syst. Sci. 44 (1992), pp. 193–219.
- [17] R. N. Bhattacharya and V. Patrangenaru, *Large sample theory of intrinsic and extrinsic sample means on manifolds – I*, Ann. Statist. 31 (2003), pp. 1–29.
- [18] S. Bigelow, *Braid groups are linear*, J. Amer. Math. Soc. 14 (2001), pp. 471–486.
- [19] P. Billingsley, *Probability and Measure*. Wiley-Interscience, 1995.
- [20] J.-C. Birget, S. Magliveras, and M. Sramka, *On public-key cryptosystems based on combinatorial group theory*, Tatra Mountains Mathematical Publications 33 (2006), pp. 137–148.
- [21] J. Birman, *Braids, Links and Mapping Class Groups*, Annals of Math. Studies. Princeton University Press, 1974.
- [22] J. S. Birman, V. Gebhardt, and J. Gonzalez-Meneses, *Conjugacy in Garside groups I: Cyclings, powers, and rigidity*, Groups Geom. Dyn. 1 (2007), pp. 221–279.
- [23] ———, *Conjugacy in Garside Groups III: Periodic braids*, J. Algebra 316 (2007), pp. 746–776.

- [24] ———, *Conjugacy in Garside groups II: Structure of the ultra summit set*, Groups Geom. Dyn. 2 (2008), pp. 13–61.
- [25] A. Bogdanov and L. Trevisan, *Average-Case Complexity*, Foundations and Trends in Theoretical Computer Science. Now Publishers Inc, 2006.
- [26] O. Bogopolski, A. Martino, O. Maslakova, and E. Ventura, *Free-by-cyclic groups have solvable conjugacy problem*, Bull. London Math. Soc. 38 (2006), pp. 787–794.
- [27] B. Bollobas, *Graph Theory: An Introductory Course*, Graduate Texts in Mathematics. Springer, 1990.
- [28] A. Borovik, A. Myasnikov, and V. Shpilrain, *Measuring sets in infinite groups*. Computational and Statistical Group Theory, Contemporary Mathematics 298, pp. 21–42. American Mathematical Society, 2002.
- [29] A. V. Borovik, A. G. Myasnikov, and V. N. Remeslennikov, *Multiplicative measures on free groups*, Int. J. Algebra Comput. 13 (2003), pp. 705–731.
- [30] ———, *Algorithmic stratification of the conjugacy problem in Miller's groups*, Int. J. Algebra Comput. 17 (2007), pp. 963–997.
- [31] ———, *The conjugacy problem in amalgamated products I: regular elements and black holes*, Int. J. Algebra Comput. 17 (2007), pp. 1301–1335.
- [32] N. Brady, T. Riley, and H. Short, *The geometry of the Word Problem for finitely generated groups*, Advanced Courses in Mathematics CRM Barcelona. Birkhauser, 2007.
- [33] M. Bridson and J. Howie, *Conjugacy of finite subsets in hyperbolic groups*, Int. J. Algebra Comput. 15 (2005), pp. 725–756.
- [34] M. Brin and C. Squier, *Groups of piecewise linear homomorphisms of the real line*, Invent. Math. 79 (1985), pp. 485–498.
- [35] R. M. Bryant, C. K. Gupta, F. Levin, and H. Y. Mochizuki, *Non-tame automorphisms of free nilpotent groups*, Commun. Algebra 18 (1990), pp. 3619–3631.
- [36] J. W. Cannon, W. J. Floyd, and W. R. Parry, *Introductory notes on Richard Thompson's groups*, L'Enseignement Mathematique 42 (1996), pp. 215–256.
- [37] G. Chartrand and O. Oellermann, *Applied and Algorithmic Graph Theory*. McGraw-Hill, New York, 1993.
- [38] K. T. Chen, R. H. Fox, and R. C. Lyndon, *Free differential calculus IV*, Ann. of Math. 71 (1960), pp. 408–422.
- [39] M. Chioldo, *Finding non-trivial elements and splittings in groups*, preprint. Available at <http://arxiv.org/abs/1002.2786>, 2010.
- [40] S. Cleary, M. Elder, A. Rechnitzer, and Taback J., *Random subgroups of Thompson's group F*, Groups Geom. Dyn. 4 (2010), pp. 91–126.
- [41] C. Coarfa, D. D. Demopoulos, A. S. M. Aguirre, D. Subramanian, and M. Y. Vardi, *Random 3-SAT: The Plot Thickens*. Proceedings of the International Conference on Constraint Programming, pp. 143–159, 2000.
- [42] D. Collins, *Relations among the squares of the generators of the braid group*, Invent. Math. 117 (1994), pp. 525–529.
- [43] S. A. Cook and D. G. Mitchell, *Finding hard instances of the satisfiability problem: a survey*, Satisfiability Problem: Theory and Applications, 35, American Mathematical Society, 1997, pp. 1–17.
- [44] S. B. Cooper, *Computability Theory*. Chapman and Hall/CRC Mathematics, 2003.
- [45] T. Coulbois, *Propriétés de Ribes-Zaleskii, topologie profinie, produit libre et généralisations*, Ph.D. thesis, Université de Paris VII, 2000.
- [46] D. F. Cowan, *A class of varieties of inverse semigroups*, J. Algebra 141 (1991), pp. 115–142.
- [47] R. Cramer and V. Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*. Advances in Cryptology – CRYPTO 1998, Lecture Notes Comp. Sc. 1462, pp. 13–25. Springer-Verlag, London, UK, 1998.
- [48] R. Crowell and R. Fox, *Introduction to Knot Theory*, Graduate Texts in Mathematics. Springer, 1984.
- [49] Cryptography and braid groups, available at <http://research.cyber.ee/~lipmaa/crypto/link/public/braid/>.
- [50] Cryptography And Groups (CRAG) C++ Library, available at <http://www.acc.stevens.edu/downloads.php>.
- [51] J. Cuesta and C. Matran, *The strong law of large numbers for k-means and best possible nets of Banach valued random variables*, Probab. Th. Rel. Fields 78 (1988), pp. 523–534.

- [52] G. d'Atri and C. Puech, *Probabilistic analysis of the subset sum problem*, Discrete Appl. Math. 4 (1982), pp. 329–334.
- [53] G. Debreu and I. Herstein, *Nonnegative square matrices*, Econometrica 21 (1953), pp. 597–607.
- [54] P. Dehornoy, *Braid groups and left distributive operations*, Trans. Amer. Math. Soc. 345 (1994), pp. 115–151.
- [55] ———, *A fast method for comparing braids*, Adv. Math. 125 (1997), pp. 200–235.
- [56] ———, *Braids and Self Distributivity*. Birkhäuser, 2000.
- [57] ———, *Braid-based cryptography*. Group theory, statistics, and cryptography, Contemporary Mathematics 360, pp. 5–33. American Mathematical Society, 2004.
- [58] ———, *Using shifted conjugacy in braid-based cryptography*. Algebraic Methods in Cryptography, Contemporary Mathematics 418, pp. 65–74. American Mathematical Society, 2006.
- [59] P. Dehornoy, I. Dynnikov, D. Rolfsen, and B. Wiest, *Why are braids orderable?*. Societe Mathematique De France, 2002.
- [60] P. Diaconis, *Group Representation in Probability and Statistics*, Lecture Notes – Monograph Series. Institute of Mathematical Statistics, 1988.
- [61] R. Diestel, *Graph Theory*, Graduate Texts in Mathematics. Springer, 2006.
- [62] C. Droms, J. Lewin, and H. Servatius, *The length of elements in free solvable groups*, Proc. Amer. Math. Soc. 119 (1993), pp. 27–33.
- [63] I. Dynnikov and B. Wiest, *On the complexity of braids*, J. Eur. Math. Soc. 9 (2007), pp. 801–840.
- [64] A. Dyubina, *Instability of the virtual solvability and the property of being virtually torsion-free for quasi-isometric groups*, Int. Math. Res. Notices 21 (2000), pp. 1097–1101.
- [65] B. Eick and D. Kahrobaei, *Polycyclic groups: a new platform for cryptology?*, preprint. Available at <http://arxiv.org/abs/math.GR/0411077>.
- [66] M. Elder, *A linear time algorithm to compute geodesics in solvable Baumslag-Solitar groups*, Illinois J. Math. 54 (2011), pp. 109–128.
- [67] M. Elder and A. Rechnitzer, *Some geodesic problems for finitely generated groups*, Groups, Complexity, Cryptology 2 (2010), pp. 223–229.
- [68] T. ElGamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE T. Inform. Theory IT-31 (1985), pp. 469–473.
- [69] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston, *Word processing in groups*. Jones and Bartlett Publishers, 1992.
- [70] A. Ershler, *Drift and entropy growth for random walks on groups*, Ann. of Prob. 31 (2003), pp. 1193–1204.
- [71] A. Eskin, D. Fisher, and K. Whyte, *Quasi-isometries and rigidity of solvable groups*, Pure Appl. Math. Quart. 3 (2007), pp. 927–947.
- [72] B. Farb and L. Mosher, *A rigidity theorem for the solvable Baumslag-Solitar groups. With an appendix by Daryl Cooper*, Invent. Math. 131 (1998), pp. 419–451.
- [73] ———, *Quasi-isometric rigidity for the solvable Baumslag-Solitar groups. II*, Invent. Math. 137 (1999), pp. 613–649.
- [74] U. Feige, A. Fiat, and A. Shamir, *Zero knowledge proofs of identity*, STOC '87: Proceedings of the nineteenth annual ACM Conference on Theory of Computing (1987), pp. 210–217.
- [75] W. Feller, *An Introduction to Probability Theory and Its Applications: Volume 2*. John Wiley & Sons, New York, 1971.
- [76] A. Fel'shtyn and E. Troitsky, *Twisted conjugacy separable groups*, preprint. Available at <http://arxiv.org/abs/math/0606764>.
- [77] R. Fenn, M. T. Greene, D. Rolfsen, C. Rourke, and B. Wiest, *Ordering the braid groups*, Pac. J. Math. 191 (1999), pp. 49–74.
- [78] R. H. Fox, *Free differential calculus I*, Ann. of Math. 57 (1953), pp. 547–560.
- [79] ———, *Free differential calculus II*, Ann. of Math. 59 (1954), pp. 196–210.
- [80] ———, *Free differential calculus III*, Ann. of Math. 64 (1956), pp. 407–419.
- [81] N. Franco and J. González-Meneses, *Conjugacy problem for braid groups and Garside groups*, J. Algebra 266 (2003), pp. 112–132.
- [82] M. Fréchet, *Les éléments aléatoires de nature quelconque dans un espace distancié*, Annales de l'Institut Henri Poincaré 10 (1948), pp. 215–310.
- [83] H. Furstenberg, *Noncommuting random products*, Trans. Amer. Math. Soc. 108 (1963), pp. 377–428.

- [84] H. Furstenberg and H. Kesten, *Products of random matrices*, Ann. Math. Statist. 31 (1960), pp. 457–469.
- [85] D. Garber, *Braid group cryptography*, preprint. Available at <http://arxiv.org/abs/0711.3941>.
- [86] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, *Probabilistic solutions of equations in the braid group*, Adv. Appl. Math. 35 (2005), pp. 323–334.
- [87] ———, *Length-based conjugacy search in the braid group*. Algebraic Methods in Cryptography, Contemp. Math. 418, pp. 75–88. Amer. Math. Soc., 2006.
- [88] M. Garey and D. Johnson, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Comput. 32 (1977), pp. 826–834.
- [89] M. Garey and J. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [90] P. Garrett, *Making, Breaking Codes: Introduction to Cryptology*. Prentice Hall, 2001.
- [91] M. Garzon and Y. Zalcstein, *The complexity of Grigorchuk groups with application to cryptography*, Theoret. Comput. Sci. 88 (1991), pp. 83–98.
- [92] V. Gebhardt, *A new approach to the conjugacy problem in Garside groups*, J. Algebra 292 (2005), pp. 282–302.
- [93] ———, *Conjugacy search in braid groups from a braid-based cryptography point of view*, Appl. Algebra Eng. Comm. 17 (2006), pp. 219–238.
- [94] I. Gelfand, *Sur un lemme de la théorie des espaces linéaires.*, Commun. Inst. Sci. Math. et Mecan., Univ. Kharkoff et Soc. Math. Kharkoff, IV. Ser. 13 (1936), pp. 35–40.
- [95] S. M. Gersten, *Isoperimetric and isodiametric functions of finite presentations*. Geometric group theory, Vol. 1 (Sussex, 1991), London Math. Soc. Lecture Note Ser. 181, pp. 79–96, 1991.
- [96] ———, *Dehn functions and l_1 -norms of finite presentations*. Algorithms and Classification in Combinatorial Group Theory, pp. 195–225. Springer, Berlin, 1992.
- [97] R. Gilman, A. G. Myasnikov, and D. Osin, *Exponentially generic subsets of groups*, Illinois J. Math. 54 (2011), pp. 371–388.
- [98] R. Gilman, A. G. Myasnikov, A. D. Miasnikov, and A. Ushakov, *Generic complexity of algorithmic problems*, In preparation.
- [99] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.
- [100] ———, *Zero-Knowledge twenty years after its invention*, preprint, available at <http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html>, 2002.
- [101] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proof systems*, J. ACM 38 (1991), pp. 691–729.
- [102] S. Goldwasser and S. Micali, *Probabilistic encryption*, J. Comput. Syst. Sci. 28 (1984), pp. 270–299.
- [103] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems*, SIAM J. Comput. 18 (1989), pp. 186–208.
- [104] M. I. Gonzalez-Vasco, S. Magliveras, and R. Steinwandt, *Group Theoretic Cryptography*. Chapman & Hall/CRC, to appear in 2012.
- [105] L. Goubin and N. T. Courtois, *Cryptanalysis of the TTM cryptosystem*. Advances in Cryptology – ASIACRYPT 2000, Lecture Notes Comp. Sc. 1976, pp. 44–57. Springer, Berlin, 2000.
- [106] M. Greendlinger, *On Dehn's algorithms for the conjugacy and word problems. With applications.*, Comm. Pure and Appl. Math. 13 (1960), pp. 641–677.
- [107] P. Grenander, *Probabilities on Algebraic Structures*, Dover Books on Mathematics. Dover Publications, 2008.
- [108] R. I. Grigorchuk, *Burnside's problem on periodic groups*, Funct. Anal. Appl. 14 (1980), pp. 41–43.
- [109] ———, *On growth in group theory*. Proceedings of the International Congress of Mathematicians, pp. 325–338, Kyoto, 1990.
- [110] ———, *On the problem of M. Day about nonelementary amenable groups in the class of finitely presented groups*, Math. Notes 60 (1996), pp. 774–775.
- [111] ———, *On the system of defining relations and the Schur multiplier of periodic groups generated by finite automata*. GROUPS St Andrews 1997, pp. 290–317. Cambridge Univ. Press, 1999.

- [112] D. Grigoriev, *On the complexity of the “wild” matrix problems, of the isomorphism of algebras and graphs*, Notes of Scientific Seminars of LOMI 105 (1981), pp. 10–17.
- [113] D. Grigoriev and I. Ponomarenko, *Homomorphic public-key cryptosystems over groups and rings*, Quaderni di Mathematica 13 (2004), pp. 305–325.
- [114] D. Grigoriev and V. Shpilrain, *Zero-knowledge authentication by the Sherlock Holmes method*, preprint. Available at <http://www.sci.ccny.cuny.edu/~shpil/papers.html>.
- [115] ———, *Authentication from matrix conjugation*, Groups Complex. Cryptol. 1 (2009), pp. 199–206.
- [116] ———, *Authentication schemes from actions on graphs, groups, or rings*, Ann. Pure Appl. Logic 162 (2010), pp. 194–200.
- [117] M. Gromov, *Groups of polynomial growth and expanding maps*, Publ. Math. IHES 53 (1981), pp. 53–73.
- [118] ———, *Infinite groups as geometric objects*. Proceedings of the International Congress of Mathematicians, 1, pp. 385–395, 1983.
- [119] ———, *Hyperbolic groups*. Essays in group theory, MSRI Publications 8, pp. 75–263. Springer, 1985.
- [120] ———, *Metric Structures for Riemannian and Non-Riemannian Spaces based on Structures Métriques des Variétés Riemanniennes*. Edited by J. LaFontaine and P. Pansu. Birkhäuser, 1999.
- [121] K. W. Gruenberg, *Residual properties of infinite soluble groups*, Proc. London Math. Soc. 7 (1957), pp. 29–62.
- [122] N. Gupta, *Free Group Rings*, Contemporary Mathematics 66. American Mathematical Society, 1987.
- [123] Y. Gurevich, *Average case completeness*, J. Comput. Syst. Sci. 42 (1991), pp. 346–398.
- [124] ———, *From invariants to canonization*. Bulletin of the European Association for Theoretical Computer Science, pp. 327–331. World Scientific, 2001.
- [125] ———, *The Challenger-Solver game: Variations on the theme of $P =?NP$* . Logic in Computer Science Column, The Bulletin of EATCS, pp. 112–121, October, 1989.
- [126] Y. Gurevich and S. Shelah, *Expected computation time for Hamiltonian Path problem*, SIAM J. Comput. 16 (1987), pp. 486–502.
- [127] J. D. Hamkins and A. G. Myasnikov, *The halting problem is decidable on a set of asymptotic probability one*, Notre Dame Journal of Formal Logic 47 (2006), pp. 515–524.
- [128] M. Hanan, *On Steiner’s problem with rectilinear distance*, SIAM J. Appl. Math. 14 (1966), pp. 255–265.
- [129] Pierre de la Harpe, *Topics in geometric group theory*. The University of Chicago Press, 2000.
- [130] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2001.
- [131] M. E. Hellman, *An overview of public key cryptography*, IEEE Communications Magazine (May 2002), pp. 42–49.
- [132] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Stat. Assoc. 58 (1963), pp. 13–30.
- [133] D. Hofheinz and R. Steinwandt, *A practical attack on some braid group based cryptographic primitives*. Advances in Cryptology – PKC 2003, Lecture Notes Comp. Sc. 2567, pp. 187–198. Springer, Berlin, 2003.
- [134] D. Hofheinz and D. Unruh, *An attack on a group-based cryptoraphic scheme*. Algebraic Methods in Cryptography, Contemporary Mathematics 418, pp. 133–140. American Mathematical Society, 2006.
- [135] J. Hughes, *A linear algebraic attack on the AAEG1 braid group cryptosystem*. The 7th Australasian Conference on Information Security and Privacy ACISP 2002, Lecture Notes Comp. Sc. 2384, pp. 176–189. Springer, Berlin, 2002.
- [136] J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, preprint. Available at <http://front.math.ucdavis.edu/0306.6032>.
- [137] R. Impagliazzo, *A personal view of average-case complexity*. Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT’95), pp. 134–147, 1995.
- [138] ———, *Computational Complexity Since 1980*. FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, Lecture Notes Comp. Sc. 3821, pp. 19–47. Springer, Berlin, 2005.
- [139] R. Impagliazzo and M. Naor, *Efficient cryptographic schemes provably as secure as subset sum*, J. Cryptology 9 (1996), pp. 199–216.

- [140] Clay Mathematical Institute, <http://www.claymath.org/prizeproblems/pvsnp.htm>.
- [141] K. Itô and H. P. McKean, Jr., *Potentials and the random walk*, Illinois J. Math. 4 (1960), pp. 119–132.
- [142] T. Jitsukawa, *Malnormal subgroups of free groups*. Computational and Statistical Group Theory, Contemporary Mathematics 298, pp. 83–96. American Mathematical Society, 2002.
- [143] V. Kaimanovich and A. M. Vershik, *Random walks on discrete groups: Boundary and entropy*, Ann. Probab. 11 (1983), pp. 457–490.
- [144] A. G. Kalka, *Representation attacks on the braid Diffie-Hellman public key encryption*, Appl. Algebra Eng. Comm. 17 (2006), pp. 257–266.
- [145] I. Kapovich and A. G. Miasnikov, *Stallings foldings and subgroups of free groups*, J. Algebra 248 (2002), pp. 608–668.
- [146] I. Kapovich, A. G. Miasnikov, P. Schupp, and V. Shpilrain, *Generic-case complexity, decision problems in group theory and random walks*, J. Algebra 264 (2003), pp. 665–694.
- [147] I. Kapovich, A. Myasnikov, P. Schupp, and V. Shpilrain, *Average-case complexity and decision problems in group theory*, Adv. Math. 190 (2005), pp. 343–359.
- [148] I. Kapovich and P. Schupp, *Genericity, the Arzhantseva-Ol'shanskii method and the isomorphism problem for one-relator groups*, Math. Ann. 331 (2005), pp. 1–19.
- [149] I. Kapovich, P. Schupp, and V. Shpilrain, *Generic properties of Whitehead's algorithm and isomorphism rigidity of random one-relator groups*, Pacific J. Math. 223 (2006), pp. 113–140.
- [150] M. I. Kargapolov and V. N. Remeslennikov, *The conjugacy problem for free solvable groups*, Algebra i Logika Sem. 5 (1966), pp. 15–25. (Russian).
- [151] A. Karlsson and F. Ledrappier, *On laws of large numbers for random walks*, Ann. Probab. 34 (2006), pp. 1693–1706.
- [152] R. M. Karp, *Reducibility among combinatorial problems*. Complexity of Computer Computations, Computer Applications in the Earth Sciences, pp. 85–103. Springer, 1972.
- [153] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.
- [154] D. Kendall, D. Barden, T. Carne, and H. Le, *Shape and Shape Theory*, Wiley Series in Probability and Statistics. Wiley, 1999.
- [155] L. G. Khachian, *A polynomial algorithm in linear programming*, Soviet Math. Doklady 20 (1979), pp. 191–194.
- [156] L. Khachiyan, *A polynomial algorithm in linear programming*, Dokl. Akad. Nauk SSSR 244 (1979), pp. 1093–1096.
- [157] O. Kharlampovich, *A finitely presented solvable group with unsolvable word problem*, Izvest. Ak. Nauk SSSR, Ser. Mat. 45 (1981), pp. 852–873.
- [158] O. Kharlampovich and M. Sapir, *Algorithmic problems in varieties*, Int. J. Algebra Comput. 5 (1995), pp. 379–602.
- [159] V. Klee and G. Minty, *How good is the simplex algorithm?*. Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969), pp. 159–175. Academic Press, 1972.
- [160] D. Knuth, J. H. Morris, and V. Pratt, *Fast pattern matching in strings*, SIAM J. Comput. 6 (1977), pp. 323–350.
- [161] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, and C. Park, *New public-key cryptosystem using braid groups*. Advances in Cryptology – CRYPTO 2000, Lecture Notes Comp. Sc. 1880, pp. 166–183. Springer, Berlin, 2000.
- [162] A. N. Kolmogorov, *La transformation de Laplace dans les espaces linéaires*, CD. Acad. Sci. Paris 200 (1935), pp. 1717–1718.
- [163] D. Krammer, *Braid groups are linear*, Ann. Math. 155 (2002), pp. 131–156.
- [164] Y. Kurt, *A new key exchange primitive based on the triple decomposition problem*, preprint. Available at <http://eprint.iacr.org/2006/378>.
- [165] J. C. Lagarias, *The $3x+1$ problem and its generalizations*, Amer. Math. Month. 92 (1985), pp. 3–23.
- [166] S. Lal and A. Chaturvedi, *Authentication schemes using braid groups*, preprint. Available at <http://arxiv.org/abs/cs/0507066>, 2005.
- [167] E. Lee, *Right-invariance: A property for probabilistic analysis of cryptography based on infinite groups*. Advances in Cryptology – Asiacrypt 2004, Lecture Notes Comp. Sc. 3329, pp. 103–118. Springer, Berlin, 2004.
- [168] S. J. Lee, *Algorithmic Solutions to Decision Problems in the Braid Group*, Ph.D. thesis, KAIST, 2000.

- [169] S. J. Lee and E. Lee, *Conjugacy classes of periodic braids*, preprint. Available at <http://front.math.ucdavis.edu/0702.5349>.
- [170] ———, *Potential weaknesses of the commutator key agreement protocol based on braid groups*. Advances in Cryptology – EUROCRYPT 2002, Lecture Notes Comp. Sc. 2332, pp. 14–28. Springer, Berlin, 2002.
- [171] L. Levin, *Average case complete problems*, SIAM J. Comput. 15 (1986), pp. 285–286.
- [172] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, Graduate texts in Computer Science. Springer, 1997.
- [173] M. Lohrey, *Word problems on compressed words*. Automata, languages and programming, Lecture Notes Comp. Sc. 3142, pp. 906–918. Springer-Verlag, Berlin, 2004.
- [174] M. Lohrey and S. Schleimer, *Efficient computation in groups via compression*. Computer Science in Russia (CSR 2007), Lecture Notes Comp. Sc. 4649, pp. 249–258. Springer-Verlag, Berlin, 2007.
- [175] J. Longrigg and A. Ushakov, *Cryptanalysis of the shifted conjugacy authentication protocol*, J. Math. Crypt. 2 (2008), pp. 107–114.
- [176] M. Lothaire, *Combinatorics on Words*. Cambridge University Press, 1997.
- [177] R. Lyndon and P. Schupp, *Combinatorial Group Theory*, Classics in Mathematics. Springer, 2001.
- [178] W. Magnus, *On a theorem of Marshall Hall*, Ann. of Math. 40 (1939), pp. 764–768.
- [179] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory*. Springer-Verlag, 1977.
- [180] M. R. Magyarik and N. R. Wagner, *A public key cryptosystem based on the word problem*. Advances in Cryptology – CRYPTO 1984, Lecture Notes Comp. Sc. 196, pp. 19–36. Springer, Berlin, 1985.
- [181] A. Mahalanobis, *A simple generalization of the ElGamal cryptosystem to non-abelian group*, Comm. Algebra 36 (2008), pp. 3878–3889.
- [182] S. W. Margolis and J. C. Meakin, *E-unitary inverse monoids and the Cayley graph of a group presentation*, J. Pure Appl. Algebra 58 (1989), pp. 45–76.
- [183] S. W. Margolis, J. C. Meakin, and J. B. Stephen, *Free objects in certain varieties of inverse semigroups*, Canadian J. Math. 42 (1990), pp. 1084–1097.
- [184] A. A. Markov, *On the impossibility of certain algorithms in the theory of associative systems*, Dokl. Akad. Nauk SSSR 55 (1947), pp. 587–590.
- [185] A. Martino, E. Turner, and E. Ventura, *The density of injective endomorphisms of a free group*, preprint.
- [186] Yu. V. Matiyasevich, *Simple examples of undecidable associative calculi*, Dokl. Akad. Nauk SSSR 173 (1967), pp. 1264–1266. English transl., Soviet Math. Dokl. 8 (1967), 555–557.
- [187] J. Matthews, *The conjugacy problem in wreath products and free metabelian groups*, Trans. Amer. Math. Soc. 121 (1966), pp. 329–339.
- [188] F. Matucci, *Cryptanalysis of the Shpilrain-Ushakov protocol for Thompson's group*, J. Cryptology 21 (2008), pp. 458–468.
- [189] G. Maze, C. Monico, and J. Rosenthal, *Public key cryptography based on semigroup actions*, Adv. Math. Comm. 1 (2007), pp. 489–507.
- [190] A. J. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.
- [191] A. G. Miasnikov and A. Rybalov, *On generically undecidable problems*, in preparation.
- [192] A. G. Miasnikov, V. Shpilrain, and A. Ushakov, *A practical attack on some braid group based cryptographic protocols*. Advances in Cryptology – CRYPTO 2005, Lecture Notes Comp. Sc. 3621, pp. 86–96. Springer, Berlin, 2005.
- [193] ———, *Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol*. Advances in Cryptology – PKC 2006, Lecture Notes Comp. Sc. 3958, pp. 302–314. Springer, Berlin, 2006.
- [194] ———, *Group-based Cryptography*, Advanced Courses in Mathematics - CRM Barcelona. Birkhäuser Basel, 2008.
- [195] A. G. Miasnikov and A. Ushakov, *Generic complexity of the conjugacy search problem in groups*, in preparation.
- [196] ———, *Random van Kampen diagrams and algorithmic problems in groups*, Groups Complex. Cryptol. 3 (2011), pp. 121–185.

- [197] A. G. Miasnikov, A. Ushakov, and Dong Wook Won, *Word problems in semigroups*, in preparation.
- [198] K. A. Mihailova, *The occurrence problem for direct products of groups*, Dokl. Akad. Nauk SSSR 119 (1958), pp. 1103–1105.
- [199] C. F. Miller III, *On group-theoretic decision problems and their classification*, Annals of Mathematics Studies 68. Princeton University Press, 1971.
- [200] ———, *Decision problems for groups – survey and reflections*. Algorithms and Classification in Combinatorial Group Theory, pp. 1–60. Springer, 1992.
- [201] J. Milnor, *Growth of finitely generated solvable groups*, J. Differ. Geom. 2 (1968), pp. 447–449.
- [202] T. Moh, *A public key system with signature and master key functions*, Comm. Algebra 27 (1999), pp. 2207–2222.
- [203] L. Mosher, M. Sageev, and K. Whyte, *Quasi-actions on trees. I. Bounded valence*, Ann. Math. 158 (2003), pp. 115–164.
- [204] N. Mosina, *Probability on graphs and groups: theory and applications*, Ph.D. thesis, Columbia University, 2009. Available at <http://www.math.columbia.edu/~thaddeus/theses/2009/mosina.pdf>.
- [205] N. Mosina and A. Ushakov, *Strong law of large numbers for metric spaces: central order*, in preparation.
- [206] E. Mourier, *Éléments aléatoires dans l'espace de Banach*, Ann. Inst. Henri Poincaré 13 (1953), pp. 159–244.
- [207] C. Mullan, *Cryptanalysing variants of Stickel's key agreement scheme*, preprint, 2010.
- [208] W. D. Munn, *Free inverse semigroups*, Proc. London Math. Soc. 29 (1974), pp. 385–404.
- [209] A. D. Myasnikov and A. G. Myasnikov, *Whitehead method and genetic algorithms*. Computational and experimental group theory, Contemporary Mathematics 349, pp. 89–114. American Mathematical Society, 2004.
- [210] A. D. Myasnikov, A. G. Myasnikov, and V. Shpilrain, *On the Andrews-Curtis equivalence*. Combinatorial and geometric group theory, Contemporary Mathematics 296, pp. 183–198. American Mathematical Society, 2002.
- [211] A. D. Myasnikov and A. Ushakov, *Length based attack and braid groups: Cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol*. Advances in Cryptology – PKC 2007, Lecture Notes Comp. Sc. 4450, pp. 76–88. Springer, 2007.
- [212] ———, *Cryptanalysis of Anshel-Anshel-Goldfeld-Lemieux key agreement protocol*, Groups Complex. Cryptol. 1 (2009), pp. 263–275.
- [213] H. Neumann, *Varieties of Groups*. Springer, 1967.
- [214] P. Novikov, *Unsolvability of the conjugacy problem in the theory of groups*, Izv. Acad. Nauk SSSR 18 (1954), pp. 485–524.
- [215] A. Yu. Ol'shanskii, *Geometry of Defining Relations in Groups*. Kluwer, 1991.
- [216] D. Osin and V. Shpilrain, *Public key encryption and encryption emulation attacks*. Computer Science in Russia – CSR 2008, Lecture Notes Comp. Sc. 5010, pp. 252–260. Springer, 2008.
- [217] S.-H. Paeng, K.-C. Ha, J. H. Kim, S. Chee, and C. Park, *New public key cryptosystem using finite non-abelian groups*. Advances in Cryptology – CRYPTO 2001, Lecture Notes Comp. Sc. 2139, pp. 470–485. Springer, Berlin, 2001.
- [218] O. Pandey, R. Pass, A. Sahai, W. Tseng, and M. Venkitasubramaniam, *Precise concurrent zero knowledge*, Eurocrypt 2008, Lecture Notes Comp. Sc. 4965 (2008), pp. 397–414.
- [219] C. Papadimitriou, *Computation Complexity*. Addison-Wesley, 1994.
- [220] W. Parry, *Growth series of some wreath products*, Trans. Amer. Math. Soc. 331 (1992), pp. 751–759.
- [221] M. Paterson and A. Razborov, *The set of minimal braids is co-NP-complete*, J. Algorithms 12 (1991), pp. 393–408.
- [222] D. Peifer, *Artin groups of extra-large type are automatic*, J. Pure Appl. Algebra 110 (1996), pp. 15–56.
- [223] G. Petrides, *Cryptanalysis of the public key cryptosystem based on the word problem on the Grigorchuk groups*. 9th IMA International Conference on Cryptography and Coding, Lecture Notes Comp. Sc. 2898, pp. 234–244. Springer, 2003.
- [224] B. J. Pettis, *On integration in vector spaces*, Trans. Amer. Math. Soc. 44 (1938), pp. 277–304.

- [225] S. Pincus, *Strong laws of large numbers for products of random matrices*, Trans. Amer. Math. Soc. 287 (1985), pp. 65–89.
- [226] W. Plandowski, *Testing equivalence of morphisms on context-free languages*. ESA 94 (Utrecht), Lecture Notes Comp. Sc. 855, pp. 460–470. Springer-Verlag, 1994.
- [227] G. Polya, *Über eine Aufgabe betreffend die Irrfahrt im Strassennetz*, Math. Ann. 84 (1921), pp. 149–160.
- [228] E. L. Post, *Recursive unsolvability of a problem of Thue*, J. Symbolic Logic 12 (1947), pp. 1–11.
- [229] G. Poupart and J. Stern, *Security Analysis of a Practical “on the fly” Authentication and Signature Generation*. Advances in Cryptology – EUROCRYPT 1998, Lecture Notes Comp. Sc. 1403, pp. 422–436. Springer, 1998.
- [230] M. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, MIT Laboratory for Computer Science (1979).
- [231] V. N. Remeslennikov, *Certain properties of Magnus embedding*, Algebra i Logika 8 (1969), pp. 72–76.
- [232] V. N. Remeslennikov and N. S. Romanovskii, *Algorithmic problems for solvable groups*. Word Problems II: The Oxford book, pp. 337–346. North-Holland, 1980.
- [233] V. N. Remeslennikov and V. G. Sokolov, *Certain properties of Magnus embedding*, Algebra i Logika 9 (1970), pp. 566–578.
- [234] J. Rhodes and B. Steinberg, *Profinite semigroups, varieties, expansions and the structure of relatively free profinite semigroups*, Internat. J. Algebra Comput. 11 (2001), pp. 627–672.
- [235] V. A. Roman'kov, *Unsolvability of the problem of endomorphic reducibility in free nilpotent groups and in free rings*, Algebra and Logic 16 (1977), pp. 310–320.
- [236] ———, *Equations in free metabelian groups*, Sib. Math. J. 20 (1979), pp. 469–471.
- [237] N. S. Romanovskii, *The occurrence problem for extensions of abelian by nilpotent groups*, Sib. Math. J. 21 (1980), pp. 170–174.
- [238] L. Ronyai, *Simple algebras are difficult*. Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 398–408, 1987.
- [239] G. Rubinshtein, *On multiple-point centers of normalized measures on locally compact metric spaces*, Siberian Math. J. 36 (1995), pp. 143–146.
- [240] D. Ruinsky, A. Shamir, and B. Tsaban, *Cryptanalysis of group-based key agreement protocols using subgroup distance functions*. Advances in Cryptology – PKC 2007, Lecture Notes Comp. Sc. 4450, pp. 61–75. Springer, 2007.
- [241] A. Rybalov, *On the strongly generic undecidability of the halting problem*, Theoret. Comput. Sci. 377 (2007), pp. 268–270.
- [242] S. Schleimer, *Polynomial-time word problems*, Comment. Math. Helv. 83 (2008), pp. 741–765.
- [243] C. P. Schnorr, *Efficient identification and signatures for smart cards*. Advances in Cryptology – CRYPTO 1989, Lecture Notes Comp. Sc. 435, pp. 239–252. Springer, 1990.
- [244] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley, 1998.
- [245] A. L. Shmel'kin, *Wreath products and group varieties*, Izvestiya AN SSSR, Ser. Mat. 29 (1965), pp. 149–176.
- [246] V. Shpilrain, *Automorphisms of F/R' groups*, Int. J. Algebra Comput. 1 (1991), pp. 177–184.
- [247] ———, *Assessing security of some group based cryptosystems*. Group theory, statistics, and cryptography, Contemporary Mathematics 360, pp. 167–177. American Mathematical Society, 2004.
- [248] ———, *Hashing with polynomials*. Information Security and Cryptology – ICISC 2006, Lecture Notes Comp. Sc. 4296, pp. 22–28. Springer, 2006.
- [249] ———, *Cryptanalysis of Stickel's key exchange scheme*. Computer Science in Russia – CSR 2008, Lecture Notes Comp. Sc. 5010, pp. 283–288. Springer, 2008.
- [250] V. Shpilrain and A. Ushakov, *Thompson's group and public key cryptography*. Applied Cryptography and Network Security – ACNS 2005, Lecture Notes Comp. Sc. 3531, pp. 151–164. Springer, 2005.
- [251] ———, *A new key exchange protocol based on the decomposition problem*. Algebraic Methods in Cryptography, Contemporary Mathematics 418, pp. 161–167. American Mathematical Society, 2006.

- [252] ———, *The conjugacy search problem in public key cryptography: unnecessary and insufficient*, Appl. Algebra Engrg. Comm. Comput. 17 (2006), pp. 285–289.
- [253] V. Shpilrain and G. Zapata, *Combinatorial group theory and public key cryptography*, Appl. Algebra Engrg. Comm. Comput. 17 (2006), pp. 291–302.
- [254] ———, *Using the subgroup membership search problem in public key cryptography*. Algebraic Methods in Cryptography, Contemporary Mathematics 418, pp. 169–179. American Mathematical Society, 2006.
- [255] ———, *Using decision problems in public key cryptography*, Groups Complex. Cryptol. 1 (2009), pp. 33–49.
- [256] H. Sibert, P. Dehornoy, and M. Girault, *Entity authentication schemes using braid word reduction*, Discrete Appl. Math. 154 (2006), pp. 420–436.
- [257] V. M. Sidelnikov, M. A. Cherepnev, and V. Y. Yaschenko, *Systems of open distribution of keys on the basis of noncommutative semigroups*, Russian Acad. Sci. Dokl. Math. 48 (1994), pp. 384–386.
- [258] I. M. Singer and J. A. Thorpe, *Lectures Notes on Elementary Topology and Geometry*, Undergraduate Texts in Mathematics. Springer-Verlag, 1967.
- [259] A. V. Skorohod, *Basic Principles and Applications of Probability Theory*. Springer, 2004.
- [260] S. Smale, *On the average number of steps of the simplex method of linear programming*, Math. Program. 27 (1983), pp. 241–262.
- [261] F. Spitzer, *Principles of Random Walk*. Springer, 2001.
- [262] M. Sramka, *On the security of Stickel's key exchange scheme*, J. Combin. Math. Combin. Comput. 66 (2008), pp. 151–159.
- [263] J. Stallings, *Topology of finite graphs*, Invent. Math. 71 (1983), pp. 551–565.
- [264] R. Steinwandt and A. Suárez Corona, *Cryptanalysis of a 2-party key establishment based on a semigroup action problem*, preprint, 2010.
- [265] E. Stickel, *A new method for exchanging secret keys*. Proceedings of the Third International Conference on Information Technology and Applications (ICITA05), Contemporary Mathematics 2, pp. 426–430. IEEE Computer Society, 2005.
- [266] D. R. Stinson, *Cryptography: Theory and Practice*, Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2005.
- [267] H. Sverdrup-Thygeson, *Strong law of large numbers for measures of central tendency and dispersion of random variables in compact metric spaces*, Ann. Stat. 9 (1981), pp. 141–145.
- [268] J. Talbot and D. Welsh, *Complexity and Cryptography: An Introduction*. Cambridge University Press, 2006.
- [269] R. L. Taylor, *Some laws of large numbers for normed linear spaces*, Ann. Math. Stat. 43 (1972), pp. 1267–1274.
- [270] J.-P. Tillich and G. Zémor, *Hashing with SL_2* . Advances in Cryptology – CRYPTO 1994, Lecture Notes Comp. Sc. 839, pp. 40–49. Springer, 1994.
- [271] N. Touikan, *A fast algorithm for Stallings' folding process*, Internat. J. Algebra Comput. 16 (2006), pp. 1031–1046.
- [272] G. S. Tseitin, *An associative system with undecidable equivalence problem*, MIAN 52 (1958), pp. 172–189.
- [273] K. Uchiyama, *Wiener's test for random walks with mean zero and finite variance*, Ann. Prob. 26 (1998), pp. 368–376.
- [274] A. Ushakov, *Fundamental Search Problems in Groups*, Ph.D. thesis, CUNY/Graduate Center, 2005.
- [275] R. Venkatesan and L. Levin, *Random instances of a graph coloring problem are hard*. Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 217–222, 1988.
- [276] A. M. Vershik, *Geometry and dynamics on the free solvable groups*, preprint. Erwin Schrödinger Institute, Vienna, 1999, pp. 1–16.
- [277] ———, *Dynamic theory of growth in groups: entropy, boundaries, examples*, Uspekhi Mat. Nauk 55 (2000), pp. 59–128.
- [278] A. M. Vershik and S. Dobrynin, *Geometrical approach to the free sovable groups*, preprint. Available at <http://arxiv.org/abs/math.GR/0405008>.
- [279] A. M. Vershik, S. Nechaev, and R. Bikbov, *Statistical properties of braid groups with application to braid groups and growth of heaps*, Commun. Math. Phys. 212 (2000), pp. 469–501.

- [280] A. M. Vershik and P. V. Sporyshev, *An estimate of the average number of steps in the simplex method, and problems in asymptotic integral geometry*, Dokl. Akad. Nauk SSSR 271 (1983), pp. 1044–1048.
- [281] B. A. F. Wehrfritz, *Two examples of soluble groups that are not conjugacy separable*, J. London Math. Soc. 2 (1973), pp. 312–316.
- [282] ———, *Another example of a soluble group that is not conjugacy separable*, J. London Math. Soc. 14 (1976), pp. 380–382.
- [283] C. M. Weinbaum, *On relators and diagrams for groups with one defining relator*, Illinois J.Math. 16 (1972), pp. 308–322.
- [284] W. Woess, *Cogrowth of groups and simple random walks*, Arch. Math. 41 (1983), pp. 363–370.
- [285] ———, *Random walks on infinite graphs and groups – a survey on selected topics*, Bull. London Math. Soc. 26 (1994), pp. 1–60.
- [286] H. Ziezold, *Expected figures and a strong law of large numbers for random elements in quasi-metric spaces*. Trans. 7th Prague Conf. Inf. Theory, Stat. Dec. Func., Random Processes A, pp. 591–602, 1977.

Abbreviations and Notation

3-SAT – Three satisfiability problem, 33	IP – Isomorphism problem, 17
AAG – Anshel-Anshel-Goldfeld protocol, 50, 179	λ -condition, 184
AAG Problem, 180	LBA – Length based attack, 168
AGL – Approximation of the geodesic length, 187	MP – Membership problem, 17, 190
AGLS – Approximation of the geodesic length in a subgroup, 187	MSP – Membership search problem, 17, 190
AveP , 126	NP , 33
AvePTime , 126	NPSPACE , 33
AveTime(t) , 126	NSPACE(f) , 32
B_n – Braid group, 56	NTIME(f) , 32
$C_G(g)$ – Centralizer of g in G , 44	NTM – Non-deterministic Turing machine, 26
$C(p)$ – Small cancelation condition, 65	P , 33
$C'(p)$ – Small cancelation condition, 65	PB_n – Pure braid group, 175
$\mathcal{C}_{\mathcal{A},f}$ – Control sequence, 134	PSPACE , 33
CNF – Conjunctive normal form, 33	PTM – Probabilistic Turing machine, 26
CP – Conjugacy problem, 16	QA – Quotient attack, 169
CSP – Conjugacy search problem, 16, 180	\mathcal{QI} – Groups with the quasi-isometric embedding property, 177
EXP , 33	\mathcal{QI}_{exp} – Groups with exponentially generic quasi-isometric embedding property, 177
\mathcal{FB} – Groups with the free basis property, 174	\mathcal{QI}_{gen} – Groups with generic quasi-isometric embedding property, 177
\mathcal{FB}_{exp} – Groups with exponentially generic free basis property, 174	\mathcal{QI}_{st} – Groups with strongly generic quasi-isometric embedding property, 177
\mathcal{FB}_{gen} – Groups with generic free basis property, 174	
\mathcal{FB}_{st} – Groups with strongly generic free basis property, 174	
Gen_δTIME(f) , 135	ρ_μ – Spherical asymptotic density, 132
GenP , 135	ρ_μ^* – Volume asymptotic density, 132
Gen_{str}P , 135	$\rho(R)$ – Spherical asymptotic density of a set R , 30
GLP – Geodesic length problem, 186	$\rho^*(R)$ – Volume asymptotic density of a set R , 30
GLSP – Geodesic length in a subgroup problem, 186	$\rho_n(R)$ – Frequency function for a set R , 30
GLSP* – Geodesic length in a subgroup problem, 186	$\rho_n^*(R)$ – Volume frequency function for a set R , 30
GPtime – Generic polynomial time, 135	RSP – Root search problem, 192
H_A – The halting set, 134	SAT – Satisfiability problem, 33
$H_{\mathcal{A},f}$, 134	SCSP – Simultaneous conjugacy search problem, 180
HCP – Hamiltonian Circuit Problem, 126	
HP – Halting problem, 25	

- SCSP* – Simultaneous conjugacy search problem relative to a subgroup, 180
SGPtime – Strong generic polynomial time, 135
SPACE(f), 32
 $Spec_k(G)$ – k -spectrum of G , 174
SSP – Subset sum problem, 33
TIME(f), 32
TM – Turing machine, 25
 $T_M(x)$ – Time function, 25
UMP – Uniform membership problem, 190
UMSP – Uniform membership search problem, 190
WP – Word problem, 15
WSP – Word search problem, 15

Index

- Abelian group, 65
Anshel-Anshel-Goldfeld protocol, 50
Aperiodic random walk, 340
Artin group, 68
Artin group of extra large type, 69
Assertion, 312
Asymptotic density, 172
Asymptotically visible property, 173
Atomic distribution, 117
atomic measure, 120
Authentication protocol, 11
Average, 332, 335

Balanced function, 128
Ball, 30
Basic random extension, 218
Benign algorithm scheme, 128
Benign fault, 128
Boltzmann distribution, 118
Bounded halting problem, 34
Braid group, 56
Braid word, 57

Cayley complex, 205
Cayley graph, 176
Cayley graph approximation, 201, 205
Center-set, 332
Centralizer, 44
Certificate, 33, 35
Chain distance, 209
Commutative group, 65
Commutativity condition, 46
Commutator, 50
Commuting subgroups, 17
Completeness of interactive proof, 355
Conjugacy decision problem, 41
Conjugacy problem, 16, 41
Conjugacy search problem, 16, 357
Conjunctive normal form, 33
Control sequence, 134
Coset representative function, 20
Cyclically reduced word, 65, 82

Decision algorithm, 27
Decision factorization problem, 47
Decision problem, 27
Decomposition search problem, 16, 43

Defining relators, 14
Dehn's algorithm, 65
Density function, 117
Depth, 209
Depth of a diagram, 201
Derivation system, 216
Descriptive complexity, 29
Diffie-Hellman key agreement, 8
Diffie-Hellman problem, 9
Disc asymptotic density, 132
Discrete logarithm problem, 9
Distribution respecting the size, 117
Distributional computational problem, 117
Double coset problem, 16, 43

Edge-cell chain, 209
ElGamal cryptosystem, 9
Execution flow, 25
Expansion factor, 8
Expected element, 329
Expected running time, 125
Exponential convergence rate, 134
Exponentially generic property, 173
Exponentially generic subset, 134
Exponentially generic upper bound, 135
Exponentially negligible subset, 134

Factorization problem, 17
Factorization search problem, 17, 46, 52
Finitely presented group, 14
Fox derivative, 67
Free abelian group, 289
Free basis, 14
Free basis property, 174
Free group, 14
Free metabelian group, 66, 289
Free solvable group, 289
Frequency, 172
Frequency function, 133
Fundamental ideal, 291

Garside normal form, 60, 188
Generator, 14
Generic upper bound, 134
Generic property, 173
Generic solution, 134
Generic subset, 131, 172

- Generic time complexity, 135
 Generically decidable, 134
 Genuinely k -dimensional random walk, 340
 Geodesic word, 286
 Graph isomorphism problem, 34
 Grigorchuk's group, 69
 Group word, 13
 Halting problem, 25
 Hamiltonian circuit, 126
 Handle, 58
 Handle free braid word, 59
 Handle reduction, 59
 Hash function, 8
 Homogeneous measure, 117
 Hypercomputer, 32
 Inner length, 183
 Integer programming problem, 34
 interactive proof of knowledge system, 355
 Intrinsic mean, 330
 Invertibility condition, 46
 Isomorphism problem, 17
 Iterative random generator, 218
 k -mean, 330
 k -spectrum, 174
 Key establishment protocol, 8
 Ko-Lee protocol, 41
 Language, 27
 Left-invariant measure, 357
 Length based attack, 168
 Linear algebra attack, 48
 Linear programming problem, 34
 Literal, 13
 Magnus embedding, 290
 Malnormal subgroup, 53
 Many to one reduction, 31
 Mean-set, 332, 359
 Mean-set attack, 355, 358
 Membership problem, 17
 Membership search problem, 17, 50
 Multiplicative distribution, 118
 Negligible subset, 131, 172
 Nielsen equivalent, 20
 Nielsen transformation, 20
 Nielsen-reduced set, 20
 Non-deterministic Turing machine, 26
 Normal form, 22, 42
NP-complete, 35
NP-hard, 35
 Permitted handle, 58
 Piece, 65, 82
 Polynomial convergence rate, 133
 Polynomial time on average, 126
 Polynomial time on average PTM, 120
 Polynomial time upper bound on average, 126
 Post correspondence problem, 34
 Presentation, 14
 Probabilistic encryption, 10
 Probabilistic Turing machine, 26, 119
 Probability mass function, 117
 Problem decidable generically in polynomial time, 135
 Problem decidable in polynomial time on average, 126
 Problem decidable strongly generically in polynomial time, 135
 Production complexity, 29
 Ptime reduction, 35
 Pure braid group, 175
 Quasi-isometric embedding, 176
 Quotient attack, 168, 189
 Random reduction, 120
 Randomized many-to-one reduction, 119
 Rarity function, 123
 Rectilinear Steiner tree, 305
 Rectilinear Steiner tree problem, 305
 Recurrent random walk, 341
 Reduced presentation, 200
 Reduced word, 13, 65, 82
 Reidemeister–Schreier rewriting process, 21
 Relative frequency, 334
 Relativized decision problem, 27
 Relator, 14
 Residual probability function, 133
 Rewriting system, 23
 Sample center-set, 335
 Sample mean-set, 335, 359
 Sampling weight, 335
 Sampling weight function, 335
 Satisfiability problem, 33, 36
 Schreier right coset function, 21
 Seminormal form, 61
 Shift search problem, 358
 Size stratification, 30
 Size compatible probability distribution, 119
 Size function, 29
 Size invariant measure, 117
 Size volume decomposition, 30
 Small cancellation group, 65
 Soundness of interactive proof, 355
 Sphere, 30
 Spherical ensemble of distributions, 119
 Spherical asymptotic density, 30
 Steiner point, 305
 Stickel's protocol, 47
 Straight line program, 309
 Stratification, 30, 171
 Stratum, 30

- Strongly generic property, 173
- Strongly generic subset, 172
- Strongly generic time complexity, 135
- Strongly generic upper bound, 135
- Strongly negligible subset, 172
- Subexponential convergence rate, 134
- Subexponential function, 138
- Subgroup-restricted conjugacy search problem, 52
- Subgroup-restricted decomposition search problem, 16
- Subset sum problem, 33
- Superpolynomial convergence rate, 133
- Superpolynomial function, 138
- Superpolynomially generic subset, 134
- Superpolynomially negligible subset, 134
- Symmetrized set of words, 65, 82
- Thompson's group F , 60
- Three satisfiability problem, 33
- Tietze transformations, 22
- Time upper bound on average, 126
- Travelling salesperson problem, 34
- Triple decomposition problem, 45
- Turing machine, 25
- Turing reduction, 31
- Twisted conjugacy problem, 93
- Uniform measure, 117
- Uniform spherical asymptotic density, 132
- van Kampen diagram, 208
- Vertex chain, 209
- Volume asymptotic density, 30, 132
- Volume ensemble of distributions, 119
- Volume frequency, 30
- Weight function, 334
- Witness, 27, 35
- Word problem, 15
- Word search problem, 15
- Zero-knowledge, 356
- Zero-knowledge interactive proof system, 356
- Zero-knowledge proof, 11, 91

This book is about relations between three different areas of mathematics and theoretical computer science: combinatorial group theory, cryptography, and complexity theory. It explores how non-commutative (infinite) groups, which are typically studied in combinatorial group theory, can be used in public-key cryptography. It also shows that there is remarkable feedback from cryptography to combinatorial group theory because some of the problems motivated by cryptography appear to be new to group theory, and they open many interesting research avenues within group theory.



In particular, a lot of emphasis in the book is put on studying search problems, as compared to decision problems traditionally studied in combinatorial group theory. Then, complexity theory, notably generic-case complexity of algorithms, is employed for cryptanalysis of various cryptographic protocols based on infinite groups, and the ideas and machinery from the theory of generic-case complexity are used to study asymptotically dominant properties of some infinite groups that have been applied in public-key cryptography so far.

This book also describes new interesting developments in the algorithmic theory of solvable groups and another spectacular new development related to complexity of group-theoretic problems, which is based on the ideas of compressed words and straight-line programs coming from computer science.



For additional information
and updates on this book, visit
www.ams.org/bookpages/surv-177

AMS on the Web
www.ams.org

