Generic-Case Complexity and Non-Commutative Cryptography

Alexander Wood

The Graduate Center, CUNY

Broad Overview

Today, we will discuss the use of **generic-case complexity** of **algorithmic problems** in groups to determine **platform groups** for use in **non-commutative cryptosystems.**

Broad Overview

- Algorithmic problems
- Worst-case and average-case complexity
- Generic-case complexity
- Non-commutative cryptography
- Platform groups for non-commutative cryptosystems
- Previous results on generic-case complexity and the conjugacy search problem in:
  - HNN-extensions and Miller's groups
  - Baumslag's groups

Computational Problems

[1] **Papadimitriou**, **Computational Complexity**, **1995**.

[2] **Myasnikov**, **Shpilrain**, **Ushakov**,
Non-commutative Cryptography and Complexity of Group-theoretic Problems.

Computational Problems

- Core topic in computer science for over a century
- Take an input, perform some number of steps, produce an output

[1] **Papadimitriou**, **Computational Complexity**, **1995**.

[2] **Myasnikov**, **Shpilrain**, **Ushakov**, **Non-commutative Cryptography and Complexity of Group-theoretic Problems**.

Computational Problems

- Core topic in computer science for over a century
- Take an input, perform some number of steps, produce an output
- *Decision problems* ask us a “yes" or “no" question

[1] **Papadimitriou**, **Computational Complexity**, **1995**.

[2] **Myasnikov**, **Shpilrain**, **Ushakov**, **Non-commutative Cryptography and Complexity of Group-theoretic Problems**.

Computational Problems

- Core topic in computer science for over a century
- Take an input, perform some number of steps, produce an output
- *Decision problems* ask us a “yes" or “no" question
- *Search problems* asks us to find a specific value[1][2]

[1] **Papadimitriou**, **Computational Complexity**, **1995**.

[2] **Myasnikov**, **Shpilrain**, **Ushakov**, **Non-commutative Cryptography and Complexity of Group-theoretic Problems**.

Computational Problems: The Setup

- X a finite alphabet

Computational Problems: The Setup

- X a finite alphabet
- X^* all words in X

Computational Problems: The Setup

- X a finite alphabet
- X^* all words in X
- Subsets of X^* are *languages* in X

Decision Problems

- X a finite alphabet
- X^* all words in X
- Subsets of X^* are *languages* in X

Decision Problems

- X a finite alphabet
- X^* all words in X
- Subsets of X^* are *languages* in X

A decision problem $\mathcal{D} = (L, U)$ for a language $L \subseteq U \subseteq X^*$ asks whether there is an algorithm $\mathcal{A}$ for a word $w \in U$ which determines whether $w \in L$.

Search Problems

- X a finite alphabet
- X^* all words in X
- Subsets of X^* are *languages* in X

Search Problems

- X a finite alphabet
- X^* all words in X
- Subsets of X^* are *languages* in X

A search problem $\mathcal{D}$ for finite alphabets X and Y and a predicate $R(x, y) \subseteq X^* \times Y^*$ asks to find $y \in Y^*$ such that $R(x, y)$ holds, given $x \in X^*$.

Algorithmic Problems in Groups

- Some classical problems were introduced by Max Dehn in the early 1900's[3].

[3] **M. Dehn**, **On the topology of three-dimensional space**, **1907**.

Algorithmic Problems in Groups

- Some classical problems were introduced by Max Dehn in the early 1900's[3].
    - The conjugacy problem

[3] **M. Dehn**, **On the topology of three-dimensional space**, **1907**.

Algorithmic Problems in Groups

- Some classical problems were introduced by Max Dehn in the early 1900's[3].
  - The conjugacy problem
  - The word problem

[3] **M. Dehn**, **On the topology of three-dimensional space**, **1907**.

The Word Problem

Word Decision Problem (WDP): Consider a finitely generated group $G = \langle X|R\rangle$. Given a word w in the generators of G, determine whether $w =_G 1$.

The Word Problem

Word Decision Problem (WDP): Consider a finitely generated group $G = \langle X | R \rangle$. Given a word w in the generators of G, determine whether $w =_G 1$.

Word Search Problem (WSP): Consider a finitely generated group $G = \langle X | R \rangle$. Let w be a word in the generators of G such that $w =_G 1$. Find a representation of w as a product of conjugates of relators from R.

The Conjugacy Problem

Conjugacy Decision Problem (CDP): Let G be a finitely generated group and let $x, y \in G$. Determine whether x and y are conjugate in G.

The Conjugacy Problem

Conjugacy Decision Problem (CDP): Let G be a finitely generated group and let $x, y \in G$. Determine whether x and y are conjugate in G.

Conjugacy Search Problem (CSP): Let G be a finitely generated group and let $x, y \in G$ such that x and y are conjugate. Find a conjugator. In other words, find an element $a \in G$ such that $x = a^{-1}ya$.

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
  - Time

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
  - Time
  - Space

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
  - Time
  - Space
- Need to know: model of computation, mode of computation, resources to be controlled, bound on controlled resource

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
  - Time
  - Space
- Need to know: model of computation, mode of computation, resources to be controlled, bound on controlled resource
  - Model: Multi-tape Turing machine

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
  - Time
  - Space
- Need to know: model of computation, mode of computation, resources to be controlled, bound on controlled resource
  - Model: Multi-tape Turing machine
  - Mode: Deterministic

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
    - Time
    - Space
- Need to know: model of computation, mode of computation, resources to be controlled, bound on controlled resource
    - Model: Multi-tape Turing machine
    - Mode: Deterministic
    - Bound: Non-decreasing function $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$

Complexity

- Complexity class of an algorithm is a method of describing the resources needed by that algorithm
  - Time
  - Space
- Need to know: model of computation, mode of computation, resources to be controlled, bound on controlled resource
  - Model: Multi-tape Turing machine
  - Mode: Deterministic
  - Bound: Non-decreasing function $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$
  - For f, there must be a multi-tape Turing machine M_f such that for any input x with size n, M computes a string $0^{f(|x|)}$ in time $T_M(x) = \mathcal{O}(n + f(n))$

Worst-Case Complexity Classes

- **TIME**($f(n)$) is the set of languages which can be decided by a multi-tape Turing machine within the time bound $f(n)$.

Worst-Case Complexity Classes

- **TIME**($f(n)$) is the set of languages which can be decided by a multi-tape Turing machine within the time bound $f(n)$.
- **NTIME**($f(n)$) is the set of languages which can be decided by nondeterministic Turing machines within the time bound $f(n)$.

Worst-Case Complexity Classes

- **TIME**$(f(n))$ is the set of languages which can be decided by a multi-tape Turing machine within the time bound $f(n)$.
- **NTIME**$(f(n))$ is the set of languages which can be decided by nondeterministic Turing machines within the time bound $f(n)$.
- **P** is the set of all languages which can be decided in polynomial time by multi-tape Turing machines. More precisely,

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathbf{TIME}(n^k).$$

Worst-Case Complexity Classes

- **TIME**$(f(n))$ is the set of languages which can be decided by a multi-tape Turing machine within the time bound $f(n)$.
- **NTIME**$(f(n))$ is the set of languages which can be decided by nondeterministic Turing machines within the time bound $f(n)$.
- **P** is the set of all languages which can be decided in polynomial time by multi-tape Turing machines. More precisely,

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathbf{TIME}(n^k).$$

- **NP** is the set of all languages which can be decided in polynomial time by nondeterministic Turing machines, i.e.,

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(n^k).$$

Deficiencies of Worst-Case Complexity

- Algorithm can be have very differently on average than it does in the worst case

Deficiencies of Worst-Case Complexity

- Algorithm can be have very differently on average than it does in the worst case
- *Example:* Hamiltonian Circuit problem is **NP**-complete but linear on average.

Deficiencies of Worst-Case Complexity

- Algorithm can be have very differently on average than it does in the worst case
- *Example:* Hamiltonian Circuit problem is **NP**-complete but linear on average.
- *Average-Case Complexity* takes into account the behavior of an algorithm on all inputs rather than just the "worst" by looking at the input distribution

Distributional Computational Problems

Definition (Probability Measure)

Let $(I, \mathcal{M})$ *be a measurable space. A probability measure on* I *is a map* $\mu : \mathcal{M} \to [0, \infty)$ *satisfying:*

(i) $\mu(\emptyset) = 0$

(ii) $\mu(I) = 1$

(iii) *If* $\{I_n\}$ *is a collection of pairwise disjoint measurable sets, then*

$$\mu\left(\bigcup_{n=1}^{\infty} I_n\right) = \sum_{n=1}^{\infty} \mu(I_n).$$

Distributional Computational Problems

Definition (Probability Measure)

Let $(I, \mathcal{M})$ be a measurable space. A probability measure on I is a map $\mu : \mathcal{M} \to [0, \infty)$ satisfying:

(i) $\mu(\emptyset) = 0$

(ii) $\mu(I) = 1$

(iii) *If $\{I_n\}$ is a collection of pairwise disjoint measurable sets, then*

$$\mu\left(\bigcup_{n=1}^{\infty} I_n\right) = \sum_{n=1}^{\infty} \mu(I_n).$$

If I is discrete (enumerable), then probability distributions μ are called atomic. ie, For a subset $S \subseteq I$,

$$\mu(S) = \sum_{x \in S} \mu(x)$$

Distributional Computational Problems

Definition

A distributional computational problem *is a pair* $(\mathcal{D}, \mu)$ *where* $\mathcal{D} = (L, I)$ *is a computational problem and* μ *is a probability measure on* I*.*

Average-Case Complexity

Let I be a discrete set with size function $s : I \rightarrow N$ and atomic probability measure μ.

Average-Case Complexity

Let I be a discrete set with size function $s : I \to N$ and atomic probability measure μ.

Definition (Linear and polynomial on μ-average functions)

A function $f : I \to \mathbb{R}^+$ is called linear on μ-average if

$$\int_I f(w)s(w)^{-1}\mu(w) < \infty.$$

A function f is called polynomial on μ-average if $f \leq p(\ell)$ for some polynomial p and some linear on μ-average function ℓ.

Average-Case Complexity (Cont.)

Average behavior of functions can be described not just as linear or polynomial but with also respect to a more general function.

Average-Case Complexity (Cont.)

Average behavior of functions can be described not just as linear or polynomial but with also respect to a more general function.

Definition (t on μ-average function)

Let $f : I \to \mathbb{R}$ and $t : \mathbb{R} \to \mathbb{R}$ be two functions. Then f is t on μ-average if $f(w) = t(\ell(x))$ for some linear on μ-average function ℓ.

Average-Case Complexity (Cont.)

The average behavior of functions can be used to define average behavior of algorithms. Let $\mathcal{D}$ be a stratified distributional algorithmic problem. Now, we let I denote the set of instances of $\mathcal{D}$.

Average-Case Complexity (Cont.)

The average behavior of functions can be used to define average behavior of algorithms. Let $\mathcal{D}$ be a stratified distributional algorithmic problem. Now, we let I denote the set of instances of $\mathcal{D}$.

Definition (Time upper bound on μ-average)

Let $\mathcal{A}$ be an algorithm. If the time function $T_{\mathcal{A}} : I \to \mathbb{N}$ has an upper bound which is t on μ-average, then we say that the algorithm has time upper bound $t(x)$ on μ-average. In particular, if $T_{\mathcal{A}}$ is polynomial on μ-average then $\mathcal{A}$ has polynomial time on μ-average.

Average-Case Complexity Classes

We can now define the following average-case complexity classes:

Average-Case Complexity Classes

We can now define the following average-case complexity classes:

- **AveP** is the class of stratified distributional problems for which there exists a polynomial time on μ-average decision algorithm.

Average-Case Complexity Classes

We can now define the following average-case complexity classes:

- **AveP** is the class of stratified distributional problems for which there exists a polynomial time on μ-average decision algorithm.
- **AveTime**(t) is the class of stratified distributional problems for which, given time bound t, there exists a decision algorithm with time upper bound t on μ-average.

Deficiencies of Average-Case Complexity

Deficiencies of Average-Case Complexity

- Sometimes an algorithm computes relatively quickly on most inputs with bad behavior on only a small number of inputs

Deficiencies of Average-Case Complexity

- Sometimes an algorithm computes relatively quickly on most inputs with bad behavior on only a small number of inputs
- Despite computing quickly on most inputs, this bad behavior can cause high worst and average case complexities for the algorithm.

Deficiencies of Average-Case Complexity

- Sometimes an algorithm computes relatively quickly on most inputs with bad behavior on only a small number of inputs
- Despite computing quickly on most inputs, this bad behavior can cause high worst and average case complexities for the algorithm.
- Can only consider decidable problems

Deficiencies of Average-Case Complexity

- Sometimes an algorithm computes relatively quickly on most inputs with bad behavior on only a small number of inputs
- Despite computing quickly on most inputs, this bad behavior can cause high worst and average case complexities for the algorithm.
- Can only consider decidable problems
- Algorithm must terminate on all inputs

Generic-Case Complexity: Idea

Generic-case complexity was first introduced in *Generic-case complexity, decision problems in group theory, and random walks* by Kapovich, Myasnikov, Schupp, and Shpilrain.

Generic-Case Complexity: Idea

Generic-case complexity was first introduced in *Generic-case complexity, decision problems in group theory, and random walks* by Kapovich, Myasnikov, Schupp, and Shpilrain.

- Computes the behavior of algorithms on “most" inputs

Generic-Case Complexity: Idea

Generic-case complexity was first introduced in *Generic-case complexity, decision problems in group theory, and random walks* by Kapovich, Myasnikov, Schupp, and Shpilrain.

- Computes the behavior of algorithms on “most" inputs
- Can consider undecidable problems

Generic-Case Complexity: Idea

Generic-case complexity was first introduced in *Generic-case complexity, decision problems in group theory, and random walks* by Kapovich, Myasnikov, Schupp, and Shpilrain.

- Computes the behavior of algorithms on “most" inputs
- Can consider undecidable problems
- It is easier to find a fast generic algorithm than it is to find an algorithm which is fast on average

Generic-Case Complexity, First Definition

Generic-Case Complexity, First Definition

- Let X be a finite alphabet, X^* the set of finite words over X

Generic-Case Complexity, First Definition

- Let X be a finite alphabet, X^* the set of finite words over X
- Let ν be a probability distribution on X^*

Generic-Case Complexity, First Definition

- Let X be a finite alphabet, X^* the set of finite words over X
- Let ν be a probability distribution on X^*
- We say that a subset $T \subset X^*$ is generic with respect to ν if $\nu(X^* \setminus T) = 0$.

Generic-Case Complexity, First Definition

- Let X be a finite alphabet, X^* the set of finite words over X
- Let ν be a probability distribution on X^*
- We say that a subset $T \subset X^*$ is generic with respect to ν if $\nu(X^* \setminus T) = 0$.
- If an algorithm $\mathcal{A}$ runs in polynomial time on all of the inputs from some subset T of X^* which is generic with respect to ν, then $\mathcal{A}$ is said to have *polynomial-time generic case complexity with respect to* ν.

GCC, First Definition - Asymptotic Density

Method of measuring our sets.

GCC, First Definition - Asymptotic Density

Method of measuring our sets.

Definition (Asymptotic density, finite alphabet version)

Let X be a finite alphabet containing at least two elements and let $(X^*)^k$ be the set of k-tuples of words on X. Define the length of any k-tuple of words $(w_1, \ldots, w_k)$ to be the sum $\sum_{i=1}^{k} w_i$, and let B_n denote the set of all k-tuples in $(X^*)^k$ of length less than or equal to n, $n \geq 0$.
For a subset $S \subseteq (X^*)^k$ define the asymptotic density $\rho(S)$ by

$$\rho(S) := \limsup_{n \to \infty} \rho_n(S)$$

where

$$\rho_n(S) := \frac{|S \cap B_n|}{|B_n|}.$$

When the limit $\lim_{n \to \infty} \rho(S)$ exists, we let $\hat{\rho}(S)$ denote $\rho(S)$.

GCC, First Definition

Definition (Generic sets, finite alphabet version)

A subset $S \subseteq (X^*)^k$ *is a generic set if* $\hat{\rho}(S) = 1$. *If* $\rho_n(S)$ *converges to* 1 *exponentially fast then* S *is said to be strongly generic.*

GCC, First Definition - Generic Performance of Algorithm

Definition (Generic and strong generic performance of a partial algorithm)

Consider a decision problem $\mathcal{D} \subseteq (X^*)^k$ *with complexity class* $\mathcal{C}$*, and let* $\mathcal{A}$ *be a correct partial algorithm for* $\mathcal{D}$*. (In other words, if* $\mathcal{A}$ *reaches a decision then that decision is correct.) Say that* $\mathcal{A}$ solves $\mathcal{D}$ with generic-case complexity $\mathcal{C}$ *if there is a generic subset* $S \subseteq (X^*)^k$ *such that for every* $\tau \in S$*,* $\mathcal{A}$ *terminates on* τ *in complexity bound* $\mathcal{C}$*. Furthermore, when S is strongly generic then* $\mathcal{A}$ solves the problem $\mathcal{D}$ with generic case complexity strongly in $\mathcal{C}$*.*

Generic-Case Complexity, Another Definition

The next definition is similar to the previous one, but does not use asymptotic density.

Generic-Case Complexity: Pseudomeasures

- In *Non-Commutative Cryptography and Complexity of Group-theoretic Problems* by Myasnikov, Shpilrain, and Ushakov, generic-case complexity is also defined in terms of generic sets

Generic-Case Complexity: Pseudomeasures

- In *Non-Commutative Cryptography and Complexity of Group-theoretic Problems* by Myasnikov, Shpilrain, and Ushakov, generic-case complexity is also defined in terms of generic sets
- Generic sets are here defined via the concept of *pseudomeasures* which “measure" the sets

Generic-Case Complexity: Pseudomeasures

Definition (Pseudomeasure)

A pseudo-measure μ on I is a function $\mu : S \to \mathbb{R}^+$ defined on a subset $S \subset \mathcal{P}(I)$ which satisfies:

Generic-Case Complexity: Pseudomeasures

Definition (Pseudomeasure)

A pseudo-measure μ on I is a function $\mu : S \to \mathbb{R}^+$ defined on a subset $S \subset \mathcal{P}(I)$ which satisfies:

1) *S contains I and is closed under disjoint union and complementation;*

Generic-Case Complexity: Pseudomeasures

Definition (Pseudomeasure)

A pseudo-measure μ on I is a function $\mu : S \to \mathbb{R}^+$ defined on a subset $S \subset \mathcal{P}(I)$ which satisfies:

1) *S contains I and is closed under disjoint union and complementation;*
2) $\mu(I) = 1$*, and*

Generic-Case Complexity: Pseudomeasures

Definition (Pseudomeasure)

A pseudo-measure μ on I is a function $\mu : S \to \mathbb{R}^+$ defined on a subset $S \subset \mathcal{P}(I)$ which satisfies:

1) *S contains I and is closed under disjoint union and complementation;*
2) *$\mu(I) = 1$, and*
3) *for any disjoint subset $A, B \in \mathcal{S}$, $\mu(A \cup B) = \mu(A) + \mu(B)$.*

More specifically, we say that a pseudo-measure μ is atomic if $\mu(Q)$ is defined for any finite subset $Q \subseteq I$.

Generic-Case Complexity: Pseudomeasures

Definition (Generic set, pseudomeasure version)

Let μ be a pseudomeasure on a set I. A subset $Q \subseteq I$ is called generic if $\mu(Q) = 1$ and is called negligible if $\mu(Q) = 0$.

GCC, First Definition - Asymptotic Density

A psuedomeasure is given by asymptotic density.

GCC, First Definition - Asymptotic Density

A psuedomeasure is given by asymptotic density.

Definition (Asymptotic density, finite alphabet version)

Let X be a finite alphabet containing at least two elements and let $(X^*)^k$ be the set of k-tuples of words on X. Define the length of any k-tuple of words $(w_1, \ldots, w_k)$ to be the sum $\sum_{i=1}^k w_i$, and let B_n denote the set of all k-tuples in $(X^*)^k$ of length less than or equal to n, $n \geq 0$.
For a subset $S \subseteq (X^*)^k$ define the asymptotic density $\rho(S)$ by

$$\rho(S) := \limsup_{n\to\infty} \rho_n(S)$$

where

$$\rho_n(S) := \frac{|S \cap B_n|}{|B_n|}.$$

When the limit $\lim_{n\to\infty} \rho(S)$ exists, we let $\hat{\rho}(S)$ denote $\rho(S)$.

Generic-Case Complexity: Pseudomeasures

Definition (Generic performance of an algorithm, pseudomeasure version)

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.

Generic-Case Complexity: Pseudomeasures - Generic Upper Bound

Let $s : I \to \mathbb{N}$ a size function on the set of inputs $I = I_{\mathcal{D}}$.

Definition (Generic upper bound)

A time function $f(n)$ *is a* generic upper bound *for* $\mathcal{A}$ *if the set*

$$H_{\mathcal{A},f} = \{w \in I : T_{\mathcal{A}}(w) \leq f(s(w))\}$$

is generic in I *with respect to the spherical asymptotic density* ρ_{μ}.

Generic-Case Complexity: A Probablistic Definition

- Kapovich provided a briefer definition of generic-case complexity in *Musings on Generic-Case Complexity* which approaches the concept from a probablistic point of view

Generic-Case Complexity: A Probablistic Definition

- Kapovich provided a briefer definition of generic-case complexity in *Musings on Generic-Case Complexity* which approaches the concept from a probablistic point of view
- Previous definitions of generic-case complexity have required first a definition of a generic set - Kapovich's definition does not.

Generic-Case Complexity: A Probablistic Definition

- Kapovich provided a briefer definition of generic-case complexity in *Musings on Generic-Case Complexity* which approaches the concept from a probablistic point of view
- Previous definitions of generic-case complexity have required first a definition of a generic set - Kapovich's definition does not.
- His definition does not require size functions.

Generic-Case Complexity: A Probablistic Definition

- Instead of considering generic subsets, Kapovich looks at the probability that certain events occur.

Generic-Case Complexity: A Probablistic Definition

- Instead of considering generic subsets, Kapovich looks at the probability that certain events occur.
- Look at the probability that an input generated by a random process terminates in time $\mathcal{O}(f(n))$.

Generic-Case Complexity: A Probablistic Definition

- Instead of considering generic subsets, Kapovich looks at the probability that certain events occur.
- Look at the probability that an input generated by a random process terminates in time $\mathcal{O}(f(n))$.
  - A *random process* is a collection $\{W(i) : i \in I\}$ of random variables for some index set I. When I is discrete, we say that this is a *discrete random process* and can denote the process by

$$W_1, W_2, \ldots, W_n, \ldots.$$

Generic-Case Complexity: A Probablistic Definition

Shpilrain's idea for the following definition is to replace the concept of a size function which measures inputs of size *n* with a random process that generates an input for the algorithm in *n* steps.

Generic-Case Complexity: A Probablistic Definition

Definition (Generic performance of an algorithm with respect to a random process)

Let Ω be the set of inputs for a partial decision algorithm $\mathcal{A}$ with values in a set U. Consider a discrete random time process $\mathcal{W} = W_1, W_2, \ldots, W_n, \ldots$ which generates an input $W_n \in \Omega$ after n steps and let f be a function such that $f(n) \geq 0$. Say that $\mathcal{A}$ has generic-case complexity less than or equal to f with respect to $\mathcal{W}$ if

$$\lim_{n \to \infty} \Pr\left[t_{\mathcal{A}}(W_n) \leq f(n)\right] = 1,$$

where $t_{\mathcal{A}}(W_n)$ denotes the time it takes for the algorithm $\mathcal{A}$ to compute on input W_n. If this limit converges exponentially fast, say that $\mathcal{U}$ has strong generic-case time complexity $\leq f$ with respect to $\mathcal{W}$.

Analysis of Generic-Case Complexity

When analyzing problems, it is important to choose the way in which we formulate the question corresponds to the definition of generic-case complexity we are using.

Analysis of Generic-Case Complexity

The original definition for generic sets is given in terms of the asymptotic density of subsets of words from some finite alphabet.

Analysis of Generic-Case Complexity

The original definition for generic sets is given in terms of the asymptotic density of subsets of words from some finite alphabet.

- Computing the asymptotic density function requires defining a length function

Analysis of Generic-Case Complexity

The original definition for generic sets is given in terms of the asymptotic density of subsets of words from some finite alphabet.

- Computing the asymptotic density function requires defining a length function
- It also requires that we are able to perform computations with B_n, sets of k-tuples of words with length at most n.

Analysis of Generic-Case Complexity

The original definition for generic sets is given in terms of the asymptotic density of subsets of words from some finite alphabet.

- Computing the asymptotic density function requires defining a length function
- It also requires that we are able to perform computations with B_n, sets of k-tuples of words with length at most n.
- Ultimately, we must choose the length function such that these computations have meaning, and the choice of length function is not always obvious.

Analysis of Generic-Case Complexity

- Some problems persist the second definition of generic sets described:

Analysis of Generic-Case Complexity

- Some problems persist the second definition of generic sets described:
  - In order to define generic sets we still are required to pick a way of measuring subsets (in this case pseudomeasure)

Analysis of Generic-Case Complexity

- Some problems persist the second definition of generic sets described:
  - In order to define generic sets we still are required to pick a way of measuring subsets (in this case pseudomeasure)
  - The choice of pseudomeasure is still not always obvious or natural

Analysis of Generic-Case Complexity

Analysis of Generic-Case Complexity

- The probablistic definition has its own deficiencies and advantages:

Analysis of Generic-Case Complexity

- The probablistic definition has its own deficiencies and advantages:
  - A deficiency: Assumes that the elements generated at each step n in the chosen random process are valid inputs for the algorithm

Analysis of Generic-Case Complexity

- The probablistic definition has its own deficiencies and advantages:
  - A deficiency: Assumes that the elements generated at each step *n* in the chosen random process are valid inputs for the algorithm
  - An advantage: does not require that we define any sort of size function, and instead just uses the time used by a random process to generate elements as their "size."

Commutative Cryptography

[4] **Whitfield Diffie and Martin Hellman**, **New Directions in Cryptography**.

Commutative Cryptography

- Whitfield Diffie and Martin Hellman revolutionized the field of cryptography by laying the groundwork for secure key exchange in their groundbreaking 1976 paper.[4]

[4] **Whitfield Diffie and Martin Hellman**, **New Directions in Cryptography**.

Commutative Cryptography

- Whitfield Diffie and Martin Hellman revolutionized the field of cryptography by laying the groundwork for secure key exchange in their groundbreaking 1976 paper.[4]
- Diffie and Hellman introduced *public key* (or *asymmetric*) *cryptography*, where two or more parties may exchange information by communicating entirely over a public channel

[4] **Whitfield Diffie and Martin Hellman**, **New Directions in Cryptography**.

Commutative Cryptography

- Whitfield Diffie and Martin Hellman revolutionized the field of cryptography by laying the groundwork for secure key exchange in their groundbreaking 1976 paper.[4]
- Diffie and Hellman introduced *public key* (or *asymmetric*) *cryptography*, where two or more parties may exchange information by communicating entirely over a public channel
- Asymmetric encryption schemes: Diffie Hellman, ElGamal, and Cramer-Shoup

[4] **Whitfield Diffie and Martin Hellman**, **New Directions in Cryptography**.

Commutative Cryptography

- Whitfield Diffie and Martin Hellman revolutionized the field of cryptography by laying the groundwork for secure key exchange in their groundbreaking 1976 paper.[4]
- Diffie and Hellman introduced *public key* (or *asymmetric*) *cryptography*, where two or more parties may exchange information by communicating entirely over a public channel
- Asymmetric encryption schemes: Diffie Hellman, ElGamal, and Cramer-Shoup
    - Use commutative groups, rely on the hardness of the *discrete logarithm problem*.

[4] **Whitfield Diffie and Martin Hellman**, **New Directions in Cryptography**.

Discrete Log Problem

Discrete Log Problem: Let G be a cyclic group and let $g \in G$ be a generator of G. The discrete logarithm problem in G is to compute $\log_g h$ for an element $h \in G$.

[5] **Peter Shor**, **Polynomial-Time Algorithms for Prime Factorization, and Discrete Logarithms on a Quantum Computer**.

Discrete Log Problem

Discrete Log Problem: Let G be a cyclic group and let $g \in G$ be a generator of G. The discrete logarithm problem in G is to compute $\log_g h$ for an element $h \in G$.

- The discrete log problem is used in many cryptosystems today because it is believed to be hard for many groups G (e.g., cyclic groups of prime order)

[5] **Peter Shor**, **Polynomial-Time Algorithms for Prime Factorization, and Discrete Logarithms on a Quantum Computer**.

Discrete Log Problem

Discrete Log Problem: Let G be a cyclic group and let $g \in G$ be a generator of G. The discrete logarithm problem in G is to compute $\log_g h$ for an element $h \in G$.

- The discrete log problem is used in many cryptosystems today because it is believed to be hard for many groups G (e.g., cyclic groups of prime order)
- Peter Shor presented an algorithm in 1994 that is able to solve the discrete logarithm in polynomial time on a quantum computer.[5]

[5] **Peter Shor**, **Polynomial-Time Algorithms for Prime Factorization, and Discrete Logarithms on a Quantum Computer**.

Non-Commutative Cryptography

- Recently, cryptosystems have been proposed which instead use non-commutative groups.

[6] **I. Anshel**, **M. Anshel**, **and D. Goldfeld**, **An algebraic method for public-key cryptography**.

[7] **Ko**, **Lee**, **et. al.**, **New public-key cryptosystem using braid groups**.

[8] **Kahrobaei**, **Khan**, **A non-commutative generalization of ElGamal Key Exchange**.

Non-Commutative Cryptography

- Recently, cryptosystems have been proposed which instead use non-commutative groups.
  - Anshel-Anshel-Goldfeld Key Exchange[6]

[6] **I. Anshel**, **M. Anshel**, **and D. Goldfeld**, **An algebraic method for public-key cryptography**.

[7] **Ko**, **Lee**, **et. al.**, **New public-key cryptosystem using braid groups**.

[8] **Kahrobaei**, **Khan**, **A non-commutative generalization of ElGamal Key Exchange**.

Non-Commutative Cryptography

- Recently, cryptosystems have been proposed which instead use non-commutative groups.
  - Anshel-Anshel-Goldfeld Key Exchange[6]
  - Non-Commutative Diffie-Hellman[7]

[6] **I. Anshel**, **M. Anshel**, **and D. Goldfeld**, **An algebraic method for public-key cryptography**.

[7] **Ko**, **Lee**, **et. al.**, **New public-key cryptosystem using braid groups**.

[8] **Kahrobaei**, **Khan**, **A non-commutative generalization of ElGamal Key Exchange**.

Non-Commutative Cryptography

- Recently, cryptosystems have been proposed which instead use non-commutative groups.
  - Anshel-Anshel-Goldfeld Key Exchange[6]
  - Non-Commutative Diffie-Hellman[7]
  - Non-Commutative ElGamal[8]

[6] **I. Anshel**, **M. Anshel**, **and D. Goldfeld**, **An algebraic method for public-key cryptography**.

[7] **Ko**, **Lee**, **et. al.**, **New public-key cryptosystem using braid groups**.

[8] **Kahrobaei**, **Khan**, **A non-commutative generalization of ElGamal Key Exchange**.

Non-Commutative Cryptography

- Recently, cryptosystems have been proposed which instead use non-commutative groups.
  - Anshel-Anshel-Goldfeld Key Exchange[6]
  - Non-Commutative Diffie-Hellman[7]
  - Non-Commutative ElGamal[8]
- The structure of these non-commutative groups causes these cryptosystems to rely on other problems for security, such as the difficulty of the conjugacy search problem.

[6] **I. Anshel**, **M. Anshel**, **and D. Goldfeld**, **An algebraic method for public-key cryptography**.

[7] **Ko**, **Lee**, **et. al.**, **New public-key cryptosystem using braid groups**.

[8] **Kahrobaei**, **Khan**, **A non-commutative generalization of ElGamal Key Exchange**.

Anshel-Anshel-Goldfeld

This protocol uses the difficulty of the word problem in some non-commutative groups as its foundation.

Anshel-Anshel-Goldfeld

Public Information: A tuple $(G, \beta, \gamma_1, \gamma_2)$, where G is a group and $\beta, \gamma_1, \gamma_2 : G \times G \to G$ are the functions

$$\beta(u, v) = u^{-1}vu \text{ (conjugation)}$$
$$\gamma_1(u, v) = u^{-1}v$$
$$\gamma_2(u, v) = v^{-1}u$$

Anshel-Anshel-Goldfeld

Public Information: A tuple $(G, \beta, \gamma_1, \gamma_2)$, where G is a group and $\beta, \gamma_1, \gamma_2 : G \times G \to G$ are the functions

$$\beta(u, v) = u^{-1}vu \text{ (conjugation)}$$
$$\gamma_1(u, v) = u^{-1}v$$
$$\gamma_2(u, v) = v^{-1}u$$

Observe that these functions satisfy the following three conditions:

1. $\beta(u, v_1 \cdot v_2) = \beta(u, v_1) \cdot \beta(u, v_2)$ for all $u, v_1, v_2 \in G$.
2. $\gamma_1(u, \beta(v, u)) = \gamma_2(v, \beta(u, v))$ for all $u, v \in G$.
3. If $x \in G$ is private, it is infeasable to determine x given $v_i \in G$ and $\beta(x, v_i)$ for $1 \leq i \leq k$.

Anshel-Anshel-Goldfeld

The Protocol:

Anshel-Anshel-Goldfeld

The Protocol:

1. Two users A and B are each publicly assigned a subgroup of G,

$$S_A = \langle s_1, s_2, \ldots, s_m \rangle,$$
$$S_B = \langle t_1, t_2, \ldots, t_n \rangle,$$

respectively.

Anshel-Anshel-Goldfeld

The Protocol:

1. Two users A and B are each publicly assigned a subgroup of G,
$$S_A = \langle s_1, s_2, \ldots, s_m \rangle,$$
$$S_B = \langle t_1, t_2, \ldots, t_n \rangle,$$
respectively.
2. A selects $a \in S_A$ and B selects $b \in S_B$. These are the users' secret keys.

Anshel-Anshel-Goldfeld

The Protocol:

1. Two users A and B are each publicly assigned a subgroup of G,

$$S_A = \langle s_1, s_2, \ldots, s_m \rangle,$$
$$S_B = \langle t_1, t_2, \ldots, t_n \rangle,$$

respectively.

2. A selects $a \in S_A$ and B selects $b \in S_B$. These are the users' secret keys.
3. A computes and transmits $\beta(a, t_i)$ for $1 \leq i \leq n$, while user B computes and transmits $\beta(b, s_i)$ for $1 \leq i \leq m$.

Anshel-Anshel-Goldfeld

The Protocol:

1. Two users A and B are each publicly assigned a subgroup of G,

$$S_A = \langle s_1, s_2, \ldots, s_m \rangle,$$
$$S_B = \langle t_1, t_2, \ldots, t_n \rangle,$$

respectively.

2. A selects $a \in S_A$ and B selects $b \in S_B$. These are the users' secret keys.
3. A computes and transmits $\beta(a, t_i)$ for $1 \leq i \leq n$, while user B computes and transmits $\beta(b, s_i)$ for $1 \leq i \leq m$.
4. A computes $\gamma_1(a, \beta(b, a))$, B computes $\gamma_2(b, \beta(a, b))$. The key κ is:

$$\kappa = \gamma_1(a, \beta(b, a)) = \gamma_2(b, \beta(a, b)) = a^{-1}b^{-1}ab.$$

AAG and the CSP

[9] **V. Shpilrain**, **A. Ushakov**,
The Conjugacy Search Problem in Public Key Cryptography: Unnecessary and Insufficient.

AAG and the CSP

- Solving the simultaneous conjugacy search problem for $a^{-1}t_ia$ and $b^{-1}s_jb$ for $1 \leq i \leq n$ and $1 \leq j \leq m$ would yield a and b, from which the secret key could be derived.

[9] **V. Shpilrain**, **A. Ushakov**, **The Conjugacy Search Problem in Public Key Cryptography: Unnecessary and Insufficient**.

AAG and the CSP

- Solving the simultaneous conjugacy search problem for $a^{-1}t_i a$ and $b^{-1}s_j b$ for $1 \leq i \leq n$ and $1 \leq j \leq m$ would yield a and b, from which the secret key could be derived.
- However, the conjugacy search problem in G does not necessarily give us a and b as words in A and B, respectively[9]

[9] **V. Shpilrain**, **A. Ushakov**, **The Conjugacy Search Problem in Public Key Cryptography: Unnecessary and Insufficient**.

AAG and the CSP

- Solving the simultaneous conjugacy search problem for $a^{-1}t_ia$ and $b^{-1}s_jb$ for $1 \leq i \leq n$ and $1 \leq j \leq m$ would yield a and b, from which the secret key could be derived.
- However, the conjugacy search problem in G does not necessarily give us a and b as words in A and B, respectively[9]
- Thus the authors explain we must also solve the membership search problem, which states that given a and $s_1, \ldots, s_m$, we must find an expression of a as a word in $s_1, \ldots, s_m$. They claim that this problem is hard in many groups.

[9] **V. Shpilrain**, **A. Ushakov**, **The Conjugacy Search Problem in Public Key Cryptography: Unnecessary and Insufficient**.

AAG and the CSP

Despite this, we would not wish for a platform group for the cryptosystem to have a fast solution for the conjugacy search problem, because it would provide an adversary with a simple attack, even if the attack might not work in every instance.

Platform Groups

It is necessary to find groups which are secure enough to serve as platforms for non-commutative cryptosystems. Shpilrain provided a collection of properties which a platform group should satisfy:[10]

[10] **Shpilrain**, **Assessing security of some group based cryptosystems**.

Platform Groups

It is necessary to find groups which are secure enough to serve as platforms for non-commutative cryptosystems. Shpilrain provided a collection of properties which a platform group should satisfy:[10]

(P1) We must have previous results regarding the conjugacy search problem in the group.

[10] **Shpilrain, Assessing security of some group based cryptosystems**.

Platform Groups

It is necessary to find groups which are secure enough to serve as platforms for non-commutative cryptosystems. Shpilrain provided a collection of properties which a platform group should satisfy:[10]

(P1) We must have previous results regarding the conjugacy search problem in the group.

(P2) In order to have efficient key extraction within the protocol, the word problem should have a "fast" solution by a deterministic algorithm.

[10] **Shpilrain**, **Assessing security of some group based cryptosystems**.

Platform Groups

It is necessary to find groups which are secure enough to serve as platforms for non-commutative cryptosystems. Shpilrain provided a collection of properties which a platform group should satisfy:[10]

(P1) We must have previous results regarding the conjugacy search problem in the group.

(P2) In order to have efficient key extraction within the protocol, the word problem should have a “fast" solution by a deterministic algorithm.

(P3) For security, the CSP should not have a “fast" (subexponential) algorithm by a deterministic algorithm.

[10] **Shpilrain**, **Assessing security of some group based cryptosystems**.

Platform Groups

It is necessary to find groups which are secure enough to serve as platforms for non-commutative cryptosystems. Shpilrain provided a collection of properties which a platform group should satisfy:[10]

(P1) We must have previous results regarding the conjugacy search problem in the group.

(P2) In order to have efficient key extraction within the protocol, the word problem should have a “fast" solution by a deterministic algorithm.

(P3) For security, the CSP should not have a “fast" (subexponential) algorithm by a deterministic algorithm.

(P4) We should not be able to recover x from $x^{-1}ax$.

[10] **Shpilrain**, **Assessing security of some group based cryptosystems**.

Previous results

This provides motivation for studying the generic-case complexity of the conjugacy search problem in various non-commutative groups.

HNN-Extensions and Miller's Groups

- Miller constructed groups for which the word problem is decidable but the conjugacy problem is undecidable in 1992.[11]

[11] **C.F. Miller III**, **Decision problems for groups**.

[12] **Shpilrain**, **Assessing security of some group based cryptosystems**.

HNN-Extensions and Miller's Groups

- Miller constructed groups for which the word problem is decidable but the conjugacy problem is undecidable in 1992.[11]
- These properties appear promising for a platform group candidate (In fact, Shpilrain suggested these groups for further consideration in 2004[12]).

[11] **C.F. Miller III**, **Decision problems for groups**.

[12] **Shpilrain**, **Assessing security of some group based cryptosystems**.

HNN-Extensions and Miller's Groups

- Miller constructed groups for which the word problem is decidable but the conjugacy problem is undecidable in 1992.[11]
- These properties appear promising for a platform group candidate (In fact, Shpilrain suggested these groups for further consideration in 2004[12]).
- However (as Shpilrain pointed out), the conjugacy problem is undecidable generally, but no results yet existed on its difficulty generically.

[11] **C.F. Miller III**, **Decision problems for groups**.

[12] **Shpilrain**, **Assessing security of some group based cryptosystems**.

HNN-Extensions and Miller's Groups

In 2007, Borovik, Myasnikov, and Remeslennikov addressed this question.[13] They show that:

[13] **Borovik**, **Myasnikov**, **Remeslennikov**,
Generic complexity of the conjugacy problem in HNN-extensions and algorithmic stratification of Miller's groups.

HNN-Extensions and Miller's Groups

In 2007, Borovik, Myasnikov, and Remeslennikov addressed this question.[13] They show that:

- The conjugacy search problem amalgamated free products and HNN-extensions of groups is generically easy even though it can be undecidable generally

[13] **Borovik**, **Myasnikov**, **Remeslennikov**, **Generic complexity of the conjugacy problem in HNN-extensions and algorithmic stratification of Miller's groups**.

HNN-Extensions and Miller's Groups

In 2007, Borovik, Myasnikov, and Remeslennikov addressed this question.[13] They show that:

- The conjugacy search problem amalgamated free products and HNN-extensions of groups is generically easy even though it can be undecidable generally
- The CSP in Miller's group is easy on most inputs, even though it is undecidable generally.

[13] **Borovik**, **Myasnikov**, **Remeslennikov**,
Generic complexity of the conjugacy problem in HNN-extensions and algorithmic stratification of Miller's groups.

HNN-extensions: Definition

HNN-extensions: Definition

- Let $H = \langle X | \mathcal{R} \rangle$ be a group

HNN-extensions: Definition

- Let $H = \langle X | \mathcal{R} \rangle$ be a group
- Let $A = \langle U_i | i \in I \rangle$ and $B = \langle V_i | i \in I \rangle$ be isomorphic subgroups of H.

HNN-extensions: Definition

- Let $H = \langle X | \mathcal{R} \rangle$ be a group
- Let $A = \langle U_i | i \in I \rangle$ and $B = \langle V_i | i \in I \rangle$ be isomorphic subgroups of H.
- Let $\phi : A \to B$ be an isomorphism given by $U_i \mapsto V_i$ for all i.

HNN-extensions: Definition

- Let $H = \langle X | \mathcal{R} \rangle$ be a group
- Let $A = \langle U_i | i \in I \rangle$ and $B = \langle V_i | i \in I \rangle$ be isomorphic subgroups of H.
- Let $\phi : A \to B$ be an isomorphism given by $U_i \mapsto V_i$ for all i.
- The *HNN-extension* of the base group H with the stable letter t and associated subgroups A and B is given by

$$G = \langle X, t | R, t^{-1} U_i t = V_i, i \in I \rangle.$$

The authors also note that G can be written also as $\langle H, t | t^{-1} A t = B, \phi \rangle$.

Reduced Forms

The *reduced form* of elements in G:

Reduced Forms

The *reduced form* of elements in G:

- Every $g \in G$ can be written

$$g = w_0 t^{\epsilon_1} w_1 \cdots t^{\epsilon_n} w_n$$

where $\epsilon_i = \pm 1$ for all i and w_i is a word in X.

Reduced Forms

The *reduced form* of elements in G:

- Every $g \in G$ can be written

$$g = w_0 t^{\epsilon_1} w_1 \cdots t^{\epsilon_n} w_n$$

  where $\epsilon_i = \pm 1$ for all i and w_i is a word in X.
- This word is called *reduced* if it contains no subwords of the form $t^{-1} w_i t$ for $w_i \in A$ or $t w_i t^{-1}$ for $w_i \in B$

Reduced Forms

The *reduced form* of elements in G:

- Every $g \in G$ can be written

$$g = w_0 t^{\epsilon_1} w_1 \cdots t^{\epsilon_n} w_n$$

  where $\epsilon_i = \pm 1$ for all i and w_i is a word in X.
- This word is called *reduced* if it contains no subwords of the form $t^{-1} w_i t$ for $w_i \in A$ or $t w_i t^{-1}$ for $w_i \in B$
- Subwords of the above form are called *pinches*

Reduced Forms

The *reduced form* of elements in G:

- Every $g \in G$ can be written

$$g = w_0 t^{\epsilon_1} w_1 \cdots t^{\epsilon_n} w_n$$

  where $\epsilon_i = \pm 1$ for all i and w_i is a word in X.
- This word is called *reduced* if it contains no subwords of the form $t^{-1} w_i t$ for $w_i \in A$ or $t w_i t^{-1}$ for $w_i \in B$
- Subwords of the above form are called *pinches*
- The *length* of the word is the number of occurrences of t_i in a reduced form of a word.

Cyclically Reduced Forms

The reduced word

$$g = ht^{\epsilon_1} s_1 \cdots t^{\epsilon_n} s_n$$

is called cyclically reduced if either:

Cyclically Reduced Forms

The reduced word

$$g = ht^{\epsilon_1}s_1 \cdots t^{\epsilon_n}s_n$$

is called cyclically reduced if either:

- If $n = 0$, then $h \in A \cup B$ or h is not conjugate in G to any element of $A \cup B$, or

Cyclically Reduced Forms

The reduced word

$$g = ht^{\epsilon_1} s_1 \cdots t^{\epsilon_n} s_n$$

is called cyclically reduced if either:

- If $n = 0$, then $h \in A \cup B$ or h is not conjugate in G to any element of $A \cup B$, or
- If $n > 0$, then either:
  - $\epsilon_1 = \epsilon_n$
  - If $\epsilon_1 = -1$, then $s_n h \notin A$
  - if $\epsilon_1 = 1$, then $s_n h \notin B$.

Unique Normal Forms

Let S_A and S_B be systems of right coset representatives of A and B in H. The normal form of an element g is a reduced form

$$g = h_0 t^{\epsilon_1} h_1 \cdots t^{\epsilon_n} h_n$$

satisfying all of following:

Unique Normal Forms

Let S_A and S_B be systems of right coset representatives of A and B in H. The normal form of an element g is a reduced form

$$g = h_0 t^{\epsilon_1} h_1 \cdots t^{\epsilon_n} h_n$$

satisfying all of following:

- $h_0 \in H$

Unique Normal Forms

Let S_A and S_B be systems of right coset representatives of A and B in H. The normal form of an element g is a reduced form

$$g = h_0 t^{\epsilon_1} h_1 \cdots t^{\epsilon_n} h_n$$

satisfying all of following:

- $h_0 \in H$
- If $\epsilon_i = -1$ then $h_i \in S_A$

Unique Normal Forms

Let S_A and S_B be systems of right coset representatives of A and B in H. The normal form of an element g is a reduced form

$$g = h_0 t^{\epsilon_1} h_1 \cdots t^{\epsilon_n} h_n$$

satisfying all of following:

- $h_0 \in H$
- If $\epsilon_i = -1$ then $h_i \in S_A$
- If $\epsilon_i = 1$ then $h_i \in S_B$.

Algorithm: Reduced Forms

Input: Words of the form $g = w_0 t^{\epsilon_1} w_1 \cdots t^{\epsilon_n} w_n$.

Algorithm 1 Reduced forms in HNN-extensions

while The word g contains a pinch $t^{\epsilon_i} w_i t^{\epsilon_{i+1}}$ **do**
  if $w_i \in A$ and $\epsilon_i = -1$ **then**
    Rewrite w_i in the generators U_j, $j \in I$, for A.
    Replace $t^{-1} w_i t$ with $\phi(w_i)$ using substitution $t^{-1} U_j t \to V_j$.
  else if $w_i \in B$ and $\epsilon_i = 1$ **then**
    Rewrite w_i in the generators V_j, $j \in I$, for B.
    Replace $t w_i t^{-1}$ with $\phi^{-1}(w_i)$ using substitution $t V_j t^{-1} \to U_j$.
  end if
end while

Algorithm: Reduced Forms

This algorithm halts in a finite number of steps with correct output whenever the Membership Search Problem is decidable for subgroups A and B.

Algorithm: Normal Forms

Input: any word g in the standard generators of G.
Let S_A and S_B be recursive sets of representatives of A and B in H.

Algorithm: Normal Forms

Input: any word g in the standard generators of G.
Let S_A and S_B be recursive sets of representatives of A and B in H.

Algorithm 3 Normal forms in HNN-extensions

while $g \in G$ is not in normal form **do**
  Replace $t^{-1}h$ with $\phi(c)t^{-1}s$, where $h = cs$, $c \in A$, and $s \in S_A$.
  Replace th with $\phi^{-1}(c)ts$, where $h = cs$, $c \in B$, and $s \in S_B$.
  Replace $t^{\epsilon}t^{-\epsilon}$ with 1.
end while

Algorithm: Normal Forms

The *Coset Representative Search Problem* asks us to find two algorithms for which, for a word $w \in F(X)$, we find a representative for Aw in S_A and Bw in S_B.

- This algorithm uses the Coset Representative Search Problem: for a word $w \in F(X)$, find a representative for Aw in S_A and Bw in S_B

Algorithm: Normal Forms

The *Coset Representative Search Problem* asks us to find two algorithms for which, for a word $w \in F(X)$, we find a representative for Aw in S_A and Bw in S_B.

- This algorithm uses the Coset Representative Search Problem: for a word $w \in F(X)$, find a representative for Aw in S_A and Bw in S_B
- Uses the Membership Search Problem, because if s_w is the representative of Aw in S_A, then $ws_w^{-1} \in A$. Applying the algorithm for the Membership Search Problem to ws_w^{-1} yields a representation of w as $w = as_w$ for $a \in A$.

Algorithm: Normal Forms

The *Coset Representative Search Problem* asks us to find two algorithms for which, for a word $w \in F(X)$, we find a representative for Aw in S_A and Bw in S_B.

- This algorithm uses the Coset Representative Search Problem: for a word $w \in F(X)$, find a representative for Aw in S_A and Bw in S_B
- Uses the Membership Search Problem, because if s_w is the representative of Aw in S_A, then $ws_w^{-1} \in A$. Applying the algorithm for the Membership Search Problem to ws_w^{-1} yields a representation of w as $w = as_w$ for $a \in A$.
- If these problems are decidable in subgroups A and B in H with respect to S_A and S_B, then this algorithm halts in finite steps with the correct output.

Algorithm: Cyclically reduced normal forms

- The authors provide an algorithm for computing cyclically reduced normal forms in G.

Algorithm: Cyclically reduced normal forms

- The authors provide an algorithm for computing cyclically reduced normal forms in G.
- It uses:

Algorithm: Cyclically reduced normal forms

- The authors provide an algorithm for computing cyclically reduced normal forms in G.
- It uses:
  - The Membership Search Problem

Algorithm: Cyclically reduced normal forms

- The authors provide an algorithm for computing cyclically reduced normal forms in G.
- It uses:
  - The Membership Search Problem
  - The Coset Representative Search Problem

Algorithm: Cyclically reduced normal forms

- The authors provide an algorithm for computing cyclically reduced normal forms in G.
- It uses:
  - The Membership Search Problem
  - The Coset Representative Search Problem
  - The Conjugacy Membership Search Problem, which takes as input $g \in H$, and asks whether g is a conjugate of an element from A or B, and if so, to find an element in A or B, respectively, which is a conjugator.

Bad Pairs

- Let $C = A \cup B$. A *bad pair* (c, g) to be an element of $C \times G$ where $c \neq 1$, $g \notin C$, and $gcg^{-1} \in C$.

Bad Pairs

- Let $C = A \cup B$. A *bad pair* (c, g) to be an element of $C \times G$ where $c \neq 1$, $g \notin C$, and $gcg^{-1} \in C$.
- The conjugacy problem is “hard" in bad pairs.

Bad Pairs

Let $c \in C \setminus \{1\}$ and $g \in G \setminus C$, where $g = hp_1 \cdots p_k$ in normal form.

Bad Pairs

Let $c \in C \setminus \{1\}$ and $g \in G \setminus C$, where $g = hp_1 \cdots p_k$ in normal form.
(c, g) is a bad pair if and only if the following system of equations has solutions $c_1, \ldots, c_{k+1} \in C$:

Bad Pairs

Let $c \in C \setminus \{1\}$ and $g \in G \setminus C$, where $g = hp_1 \cdots p_k$ in normal form.
(c, g) is a bad pair if and only if the following system of equations has solutions $c_1, \ldots, c_{k+1} \in C$:

$$B_{c,b} = \begin{cases} p_k c p_k^{-1} & = c_1 \\ p_{k-1} c_1 p_{k-1}^{-1} & = c_2 \\ & \vdots \\ p_1 c_{k-1} p_1^{-1} & = c_k \\ h c_k h^{-1} & = c_{k+1}. \end{cases}$$

Solutions To The System of Equations

Let g and g' be elements in G with normal forms $g = hp_1 \cdots p_k$ and $g' = h'p'_1 \cdots p'_k$. The equation $gc = c'g'$ has solution $c, c' \in C$ if and only if the following system of equations in $c_1, c_2, \ldots, c_k$ has a solution in C:

$$S_{g,g'} = \begin{cases} p_k c & = c_1 p'_k \\ p_{k-1} c_1 & = c_2 p'_{k-1} \\ & \vdots \\ p_1 c_{k-1} & = c_k p'_1 \\ h c_k & = c' h'. \end{cases}$$

The *principal system of equations* is comprised of the first k equations from $S_{g,g'}$ and is denoted by $PS_{g,g'}$. Let $E_{g,g'}$ denote the set of all elements $c \in C$ such that $PS_{g,g'}$ has a solution.

The Black Hole

The *Black Hole* of the conjugacy problem in G is given by

$$\mathbb{BH} = N_G^*(C) = \{g | C^g \cap C \neq 1\},$$

where a bad pair (c, g) satisfies $c \in Z_g(C) = \{c \in C | c^{g^{-1}} \in C\}$ and $g \in N_G^*(C) \setminus C$.

The Black Hole

The *Black Hole* of the conjugacy problem in G is given by

$$\mathbb{BH} = N_G^*(C) = \{g | C^g \cap C \neq 1\},$$

where a bad pair (c, g) satisfies $c \in Z_g(C) = \{c \in C | c^{g^{-1}} \in C\}$ and $g \in N_G^*(C) \setminus C$.

- Elements in the black hole $\mathbb{BH}$ are called *singular*

The Black Hole

The *Black Hole* of the conjugacy problem in G is given by

$$\mathbb{BH} = N_G^*(C) = \{g | C^g \cap C \neq 1\},$$

where a bad pair (c, g) satisfies $c \in Z_g(C) = \{c \in C | c^{g^{-1}} \in C\}$ and $g \in N_G^*(C) \setminus C$.

- Elements in the black hole $\mathbb{BH}$ are called *singular*
- Elements outside of $\mathbb{BH}$ are called *regular*.

When is an item regular?

- uMv is a *G-shift* of M, where M is a subset of G and $u, v \in G$.

When is an item regular?

- *uMv* is a *G-shift* of *M*, where *M* is a subset of *G* and $u, v \in G$.
- Let $SI(\mathcal{M}, G)$, for a collection $\mathcal{M}$ of subsets of *G*, denote the least set of subsets of *G* such that:

When is an item regular?

- *uMv* is a *G-shift* of *M*, where *M* is a subset of *G* and $u, v \in G$.
- Let $SI(\mathcal{M}, G)$, for a collection $\mathcal{M}$ of subsets of *G*, denote the least set of subsets of *G* such that:
    - $SI(\mathcal{M}, G)$ contains $\mathcal{M}$
    - it is closed under both *G*-shifts and finite intersections.

When is an item regular?

- uMv is a *G-shift* of M, where M is a subset of G and $u, v \in G$.
- Let $SI(\mathcal{M}, G)$, for a collection $\mathcal{M}$ of subsets of G, denote the least set of subsets of G such that:
  - $SI(\mathcal{M}, G)$ contains $\mathcal{M}$
  - it is closed under both G-shifts and finite intersections.
- $Sub(C)$ is the set of all finitely generated subgroups of C.

When is an item regular?

- uMv is a *G-shift* of M, where M is a subset of G and $u, v \in G$.
- Let $SI(\mathcal{M}, G)$, for a collection $\mathcal{M}$ of subsets of G, denote the least set of subsets of G such that:
  - $SI(\mathcal{M}, G)$ contains $\mathcal{M}$
  - it is closed under both G-shifts and finite intersections.
- $Sub(C)$ is the set of all finitely generated subgroups of C.
- The Cardinality Search Problem on $SI(Sub(C), H)$ takes a set $D \in SI(Sub(C), H)$ as input and asks us to determine whether D is empty, finite, or infinite. If D is finite and nonempty, it asks us to list all elements of D.

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.
- Criterion for determining if an element is regular in a HNN-extension $G = \langle H, t | t^{-1}At = B \rangle$:

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.
- Criterion for determining if an element is regular in a HNN-extension $G = \langle H, t | t^{-1}At = B\rangle$:
  - H allows algorithms for the search membership problem in H,

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.
- Criterion for determining if an element is regular in a HNN-extension $G = \langle H, t | t^{-1}At = B \rangle$:
  - H allows algorithms for the search membership problem in H,
  - the coset representaitve search problem in H,

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.
- Criterion for determining if an element is regular in a HNN-extension $G = \langle H, t | t^{-1}At = B \rangle$:
  - H allows algorithms for the search membership problem in H,
  - the coset representaitve search problem in H,
  - the cardinality search problem for $SI(Sub(C), H)$ in H

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.
- Criterion for determining if an element is regular in a HNN-extension $G = \langle H, t | t^{-1}At = B \rangle$:
  - H allows algorithms for the search membership problem in H,
  - the coset representaitve search problem in H,
  - the cardinality search problem for $SI(Sub(C), H)$ in H
  - the Membership Problem for $N_H^*(C)$ in H.

Regular Element Criterion

- If the Cardinality Search Problem in decidable in $SI(Sub(C), H)$, then $E_{g,g'}$ can be found effectively when given g and g'.
- Criterion for determining if an element is regular in a HNN-extension $G = \langle H, t | t^{-1}At = B \rangle$:
  - H allows algorithms for the search membership problem in H,
  - the coset representaitve search problem in H,
  - the cardinality search problem for $SI(Sub(C), H)$ in H
  - the Membership Problem for $N_H^*(C)$ in H.

Then, an algorithm exists which decides whether or not a given element in G is regular or not.

The CSP for regular elements in HNN-extensions

Theorem

Consider a group G, where G is an HNN-extension of a finitely presented group H. Say $G = \langle H, t | t^{-1}At = B \rangle$. Let A and B be two finitely generated subgroups of G. Assume the group H allows algorithms for the Word Problem in H, the Search Membership Problem for A and B in H, the Coset Representative Search Problem for subgroups A and B in H, and the Cardinality Search Problem for $SI(Sub(C), H)$ in H. Then, the Conjugacy Search Problem is decidable in G for arbitrary pairs (g, u), where g has a cyclically reduced regular normal form of non-zero length and $u \in G$.

Miller's Group

Miller's groups are constructed via HNN-extensions. Let H be a finitely presented group given in terms of generators and relators as

$$H = \langle s_1, \ldots, s_n | R_1, \ldots, R_m \rangle.$$

Miller's Group

Miller's groups are constructed via HNN-extensions. Let H be a finitely presented group given in terms of generators and relators as

$$H = \langle s_1, \ldots, s_n | R_1, \ldots, R_m \rangle.$$

Miller's Group

Miller's groups are constructed via HNN-extensions. Let H be a finitely presented group given in terms of generators and relators as

$$H = \langle s_1, \ldots, s_n | R_1, \ldots, R_m \rangle.$$

The *Miller Group of H*, denoted $G(H)$, is constructed with generators

$$q, s_1, \ldots, s_n, t_1, \ldots, t_m, d_1, \ldots, d_n$$

and relators

$$\begin{aligned} t_i^{-1} q t_i &= q R_i, \\ t_i^{-1} s_j t_i &= s_j, \\ d_j^{-1} q d_j &= s_j^{-1} q s_j, \\ d_k^{-1} s_j d_k &= s_j. \end{aligned}$$

Miller's Group

- Miller's group is constructed as an HNN-extension so the authors can describe the regular form of its elements.

Miller's Group

- Miller's group is constructed as an HNN-extension so the authors can describe the regular form of its elements.
- There is a cubic time algorithm in the length of $|g|$ which finds the normal form of g, and a cubic time algorithm which finds the cyclically reduced normal form of g.

Miller's Group

- Miller's group is constructed as an HNN-extension so the authors can describe the regular form of its elements.
- There is a cubic time algorithm in the length of $|g|$ which finds the normal form of g, and a cubic time algorithm which finds the cyclically reduced normal form of g.
- However, the black hole of $G(H)$ is equal to $G(H)$.

Miller's Group

- Miller's group is constructed as an HNN-extension so the authors can describe the regular form of its elements.
- There is a cubic time algorithm in the length of $|g|$ which finds the normal form of g, and a cubic time algorithm which finds the cyclically reduced normal form of g.
- However, the black hole of $G(H)$ is equal to $G(H)$.
- This means no elements are regular!

Miller's Group: The Strong Black Hole

Miller's Group: The Strong Black Hole

- The conjugacy problem is still easy on most elements.

Miller's Group: The Strong Black Hole

- The conjugacy problem is still easy on most elements.
- The authors call these elements *weakly regular*, and the elements on which the CSP is hard *strongly singular*.

Miller's Group: The Strong Black Hole

- The conjugacy problem is still easy on most elements.
- The authors call these elements *weakly regular*, and the elements on which the CSP is hard *strongly singular*.
- Strongly singular elements lie in the *strong black hole* of $G(H)$, $\mathbb{SBH}(G)$

Miller's Group: The Strong Black Hole

- The conjugacy problem is still easy on most elements.
- The authors call these elements *weakly regular*, and the elements on which the CSP is hard *strongly singular*.
- Strongly singular elements lie in the *strong black hole* of $G(H)$, $\mathbb{SBH}(G)$
- The authors provide *conjugacy criterion* for weakly regular elements.

The CSP in Miller's Group

- If the word problem is undecidable in a finitely presented group H, then the conjugacy problem is undecidable in $G(H)$.

The CSP in Miller's Group

- If the word problem is undecidable in a finitely presented group H, then the conjugacy problem is undecidable in $G(H)$.
- The strong black hole of G is strongly negligible in G

The CSP in Miller's Group

- If the word problem is undecidable in a finitely presented group H, then the conjugacy problem is undecidable in $G(H)$.
- The strong black hole of G is strongly negligible in G
- The Conjugacy Search Problem is decidable in cubic time for all weakly regular elements in $G(H)$.

The CSP in Miller's Group

- If the word problem is undecidable in a finitely presented group H, then the conjugacy problem is undecidable in $G(H)$.
- The strong black hole of G is strongly negligible in G
- The Conjugacy Search Problem is decidable in cubic time for all weakly regular elements in $G(H)$.
- Thus, the generic-case complexity of the conjugacy search problem in $G(H)$ is easy, despite the fact that it is undecidable in general.

Proof Sketch: The SBH in G(H) is strongly negligible

- Define the sphere of radius k in a free group F to be $S_k = \{w \in F \mid |w| = k\}$. For a subset R of F, define the function f_k by

$$f_k(R) = \frac{|R \cap S_k|}{|S_k|}.$$

Proof Sketch: The SBH in G(H) is strongly negligible

- Define the sphere of radius k in a free group F to be $S_k = \{w \in F \mid |w| = k\}$. For a subset R of F, define the function f_k by

$$f_k(R) = \frac{|R \cap S_k|}{|S_k|}.$$

- The asymptotic density $\rho(R)$ is given by

$$\rho(R) = \limsup_{k \to \infty} f_k(R).$$

Proof Sketch: The SBH in G(H) is strongly negligible

- Define the sphere of radius k in a free group F to be $S_k = \{w \in F \| |w| = k\}$. For a subset R of F, define the function f_k by

$$f_k(R) = \frac{|R \cap S_k|}{|S_k|}.$$

- The asymptotic density $\rho(R)$ is given by

$$\rho(R) = \limsup_{k \to \infty} f_k(R).$$

- Recall that R is called generic if $\rho(R) = 1$ and negligible if the asymptotic density of its complement is 1. If there is a positive constant $\delta < 1$ such that $1 - \delta^k < f_k(R) < 1$ for all k greater than some constant K, then R is *strongly generic*. Similarly, R is *strongly negligible* if its complement is strongly generic.

Proof Sketch: The SBH in G(H) is strongly negligible

The authors prove that the strong black hole of G is strongly negligible:

Theorem

Let

$$H = \langle s_1, \ldots, s_n | R_1, \ldots, R_m \rangle$$

be a finitely presented group. Let $G(H)$ be the Miller's group of H. Let $m > 1$. Then, $\mathbb{SBH}(G)$ is strongly negligible, and for $k > 1$,

$$f_k(\mathbb{SBH}(G)) < \left(\frac{n+1}{n+m}\right)^{k-1}$$

Proof Sketch: The SBH in G(H) is strongly negligible

Let G_k, B_k, and P_k denote the set of all elements with length k in G, $F(S,q)$, and $F(T,D)$. Because $l(g) = l(u) + l(f)$ where $g = uf$ such that $u \in F(T,D)$ and $f \in F(S,q)$, then $|G_k| = |P_k| + |P_{k-1}||B_1| + \cdots + |B_k|$. Thus, for $m > 1$,

$$
\begin{aligned}
f_k(\mathbb{SBH}(G)) &= \frac{|B_k|}{|G_k|} \\
&< \frac{|B_k|}{|P_k|} \\
&= \frac{(2n+2)(2n+1)^{k-1}}{(2n+2m)(2n+2m-1)^{k-1}} \\
&< \left(\frac{n+1}{n+m}\right)^{k-1}.
\end{aligned}
$$

Baumslag's Group

- In *Conjugacy in Baumslag's Group, Generic Case Complexity, and Division in Power Circuits*, Diekert, Myasnikov, and Weiss study the conjugacy problem in:

Baumslag's Group

- In *Conjugacy in Baumslag's Group, Generic Case Complexity, and Division in Power Circuits*, Diekert, Myasnikov, and Weiss study the conjugacy problem in:
  - the Baumslag-Solitar group $\mathbf{BS}_{1,2}$

Baumslag's Group

- In *Conjugacy in Baumslag's Group, Generic Case Complexity, and Division in Power Circuits*, Diekert, Myasnikov, and Weiss study the conjugacy problem in:
  - the Baumslag-Solitar group $\mathbf{BS}_{1,2}$
  - Baumslag's group $\mathbf{G}_{1,2}$, an HNN-extension of the Baumslag-Solitar group.

Baumslag's Group

- In *Conjugacy in Baumslag's Group, Generic Case Complexity, and Division in Power Circuits*, Diekert, Myasnikov, and Weiss study the conjugacy problem in:
  - the Baumslag-Solitar group $\mathbf{BS}_{1,2}$
  - Baumslag's group $\mathbf{G}_{1,2}$, an HNN-extension of the Baumslag-Solitar group.
- They show CSP is generically polynomial in Baumslag's group but the average-case complexity is non-elementary.

The Baumslag Group: Definition

- The Baumslag-Solitar group is given in terms of generators and relations by

$$\mathbf{BS}_{1,2} = \langle a, t | tat^{-1} = a^2 \rangle.$$

The Baumslag Group: Definition

- The Baumslag-Solitar group is given in terms of generators and relations by

$$\mathbf{BS}_{1,2} = \langle a, t | tat^{-1} = a^2 \rangle.$$

- The Baumslag group is given by

$$\mathbf{G}_{1,2} = \langle a, b | bab^{-1}a = a^2bab^{-1} \rangle.$$

Baumslag's Group: CSP

- The WSP and CSP have low complexity in $\mathbf{BS}_{1,2}$.

Baumslag's Group: CSP

- The WSP and CSP have low complexity in $\mathbf{BS}_{1,2}$.
- The WSP and CSP in $\mathbf{G}_{1,2}$ do not have low complexity.

Baumslag's Group: CSP

- The WSP and CSP have low complexity in $\mathbf{BS}_{1,2}$.
- The WSP and CSP in $\mathbf{G}_{1,2}$ do not have low complexity.
- The CSP in Baumslag's group is generically solvable in polynomial time.

Baumslag's Group

The authors rewrite Baumslag's group with a third generator:

Baumslag's Group

The authors rewrite Baumslag's group with a third generator:

$$\mathbf{G}_{1,2} = \langle a, b, t | tat^{-1} = a^2, bab^{-1} = t \rangle.$$

Baumslag's Group

The authors rewrite Baumslag's group with a third generator:

$$\mathbf{G}_{1,2} = \langle a, b, t | tat^{-1} = a^2, bab^{-1} = t \rangle.$$

They give the elements of this group as their β-*factorizations*, which is a word

$$z = \gamma_0 \beta_1 \gamma_1 \cdots \beta_k \gamma_k$$

where $\beta_i \in \{b, \bar{b}\}$ and $\gamma_i \in \{a, \bar{a}, t, \bar{t}\}^*$, with the length of z given by $l(z) = k$.

Britton-reduced forms

- Words given in this form can be *Britton-reduced*, which means no *Britton-reductions* are possible.

Britton-reduced forms

- Words given in this form can be *Britton-reduced*, which means no *Britton-reductions* are possible.
- A word x is called *cyclically* Britton-reduced if xx is Britton-reduced

Britton-reduced forms

- Words given in this form can be *Britton-reduced*, which means no *Britton-reductions* are possible.
- A word x is called *cyclically* Britton-reduced if xx is Britton-reduced
- $\hat{x}$ denotes a cyclically Britton-reduced form of x.

Britton Reductions Algorithm

Britton reductions are described in the following algorithm.

Algorithm 4 Britton Reductions

```
for Some factor βγβ̄ with γ ∈ {a, ā, t, t̄}* do
    if β = b and γ = a^ℓ in BS_{1,2} for some ℓ ∈ ℤ then
        Replace bγb̄ with t^ℓ
    end if
    if β = b̄ and γ = t^ℓ in BS_{1,2} for some ℓ ∈ ℤ then
        Replace b̄γb with a^ℓ
    end if
end for
```

CSP in Baumslag's Group

- The word problem in $\mathbf{G}_{1,2}$ is decidable in cubic time.

CSP in Baumslag's Group

- The word problem in $\mathbf{G}_{1,2}$ is decidable in cubic time.
- Let $x, y \in \{a, \bar{a}, b, \bar{b}\}^*$. The authors show that for inputs x and y,

CSP in Baumslag's Group

- The word problem in $\mathbf{G}_{1,2}$ is decidable in cubic time.
- Let $x, y \in \{a, \bar{a}, b, \bar{b}\}^*$. The authors show that for inputs x and y,
  - The CSP can be carried out in time $\mathcal{O}(n^4)$ whenever $\ell(\hat{x}) > 0$

CSP in Baumslag's Group

- The word problem in $\mathbf{G}_{1,2}$ is decidable in cubic time.
- Let $x, y \in \{a, \bar{a}, b, \bar{b}\}^*$. The authors show that for inputs x and y,
  - The CSP can be carried out in time $\mathcal{O}(n^4)$ whenever $\ell(\hat{x}) > 0$
  - Inputs such that $\ell(\hat{x}) = 0$ form a strongly negligible set.

Acknowledgements

Many thanks to my mentor, Professor Delaram Kahrobaei, as well as the rest of the second exam committee – Professors Xiangdong Li, Vladimir Shpilrain, Ben Fine, and Xiaowen Zhang.
I appreciate your time and your generous support.

References

[AAG99] I. Anshel, M. Anshel, and D. Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters 6 (1999), 287-291.

[BMR07] A. Borovik, A. Myasnikov, and V. Remelennikov, *Generic Complexity of the Conjugacy Problem in HNN-extensions and algorithmic stratification of Miller's Groups*, Internat. J. Algebra Comput., Volume 17, Issue 5-6, p.963–997 (2007).

[D07] M. Dehn, *On the topology of three-dimensional space*, Papers on Group Theory and Topology, Springer-Verlag (1987) pp. 86-126. Originally published in German in 1907.

References

[DMW14] V, Diekert, A. Myasnikov, and A. Weiss, *Conjugacy in Baumslag's Group, Generic Case Complexity, and Division in Power Circuits*, LATIN 2014: Theoretical Informatics, Volume 8392 of the series Lecture Notes in Computer Science pp 1-12.

[DH76] Whitfield Diffie and Martin Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory IT-22 (1976), no. 6, 644-654.
Proceeding of IEEE, Pages: 1-5 (2006)

[KK06] D. Kahrobaei and B. Khan *A Non-Commutative Generalization of the ElGamal Key Exchange using Polycyclidistributionc Groups*,

References

[KL15] Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography*, CRC Press, 2015.

[KL00] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. S. Kang, and C. Park, *New public-key cryptosystem using braid groups*, Advances in cryptology CRYPTO 2000 (Santa Barbara, CA), Lecture Notes in Comp. Sc., vol. 1880 (2000), 166-183.

[MSU11] A. Myasnikov, V. Shpilrain, A. Ushakov, *Non-commutative Cryptography and Complexity of Group-theoretic Problems*, Mathematical Surveys and Monographs, Vol. 177, Providence, Rhode Island (2011).

References

[P94] C. H. Papadimitriou, *Computational Complexity*, Addison Wesley Longman, Reading, Massachusetts, 1995.

[S97] Peter W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM Journal on Computing, Volume 26 Issue 5, Oct. 1997, 1484-1509.

[S04] V. Shpilrain, *Assessing security of some group based cryptosystems*, 2003; arXiv:math/0311047.