What is new in algorithmic group theory?

Alexei Miasnikov (McGill University)

Institute of Mathematics, Omsk, August, 2009

Main topics

I. Evolution of algorithmic problems (in groups).
II. Generic complexity: typical versus worst and average.
III. Rewriting: finite, infinite , or else?
IV. New data structures in algebra: invasion of Computer Science.
V. Group-based cryptography: is it real or fake?
VI. Outstanding open problems.
VII. Experimental Math: what is this?

I. Evolution of algorithmic problems

Plan:

- Golden Age
- Unsettling times
- Decidable-undecidable
- Complexity
- Tractable problems
- Uniform problems
- Search problems and enumerating procedures
- Distributional problems
- Generic case complexity
- Change of the paradigm: quest for hard instances

Golden age: the beginning

Dehn (1910-12):

- Find an algorithm to solve the *word problem (WP)* in groups.
- Find an algorithm to solve the *conjugacy problem (CP)* in groups.
- Find and algorithm to solve the *membership problem (MP)* in groups.

Tietze (1910):

- Find an algorithm to solve the *isomorphism problem (IP)* in groups.

Common viewpoint: everything is decidable.

Representing elements

Let G be a group with a finite generating set X.

Any word w in the alphabet $X \cup X^{-1}$ represents an element in G.

Obviously, every element in G can be represented by a *freely reduced* word in $X \cup X^{-1}$ (no subwords xx^{-1} or $x^{-1}x$).

$F(X) =$ the set of all *freely reduced* words in $X \cup X^{-1}$,

$F(X)$ forms a free group with basis X, where multiplication is concatenation of two words with the subsequent free reduction.

Representing elements

There are some other ways to represent elements in a more compact way (compression) :

- the word $w = xx \dots x$ (100 times) can be represented as x^{100}.
- $w = \phi^{100}(u)$, where $u \in F(X)$ and ϕ is a fixed substitution $x \to v_x$ ($\phi \in End(F(X))$.
- w can be represented by some (simple) programm P_w to compute w.

Representing elements

There are some other ways to represent elements in a more compact way (compression) :

- the word $w = xx \dots x$ (100 times) can be represented as x^{100}.
- $w = \phi^{100}(u)$, where $u \in F(X)$ and ϕ is a fixed substitution $x \to v_x$ ($\phi \in End(F(X))$.
- w can be represented by some (simple) programm P_w to compute w.

Representing elements

There are some other ways to represent elements in a more compact way (compression) :

- the word $w = xx \dots x$ (100 times) can be represented as x^{100}.
- $w = \phi^{100}(u)$, where $u \in F(X)$ and ϕ is a fixed substitution $x \to v_x$ ($\phi \in End(F(X))$.
- w can be represented by some (simple) programm P_w to compute w.

Presentations of groups

The group G can be described as a quotient of $F(X)$, $G \simeq F(X)/N$.

If N, as a normal subgroup, is generated by a subset $R \subseteq F(X)$, then G can be described by a pair $\langle X \mid R \rangle$ - a presentation of G.

In the case when R is finite (or recursively enumerable) one can view $\langle X \mid R \rangle$ as an effective way to describe G.

This description comes from topological considerations. It is simple and nice.

The fundamental problems: precise formulations

Let G be given by a finite presentation $G = \langle X \mid R \rangle$.

- WP in G: given $w \in F(X)$ decide if $w = 1$ in G.
- CP in G: given two words $u, v \in F(X)$ decide if u and v are conjugate in G.
- MP in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ decide if $w \in H$.
- IP in a given class $\mathcal{C}$ of presentations: given two finite (or recursive) presentations in $\mathcal{C}$ decide if the groups, defined by the presentations, are isomorphic or not.

The fundamental problems: precise formulations

Let G be given by a finite presentation $G = \langle X \mid R \rangle$.

- WP in G: given $w \in F(X)$ decide if $w = 1$ in G.
- CP in G: given two words $u, v \in F(X)$ decide if u and v are conjugate in G.
- MP in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ decide if $w \in H$.
- IP in a given class $\mathcal{C}$ of presentations: given two finite (or recursive) presentations in $\mathcal{C}$ decide if the groups, defined by the presentations, are isomorphic or not.

The fundamental problems: precise formulations

Let G be given by a finite presentation $G = \langle X \mid R \rangle$.

- WP in G: given $w \in F(X)$ decide if $w = 1$ in G.
- CP in G: given two words $u, v \in F(X)$ decide if u and v are conjugate in G.
- MP in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ decide if $w \in H$.
- IP in a given class $\mathcal{C}$ of presentations: given two finite (or recursive) presentations in $\mathcal{C}$ decide if the groups, defined by the presentations, are isomorphic or not.

The fundamental problems: precise formulations

Let G be given by a finite presentation $G = \langle X \mid R \rangle$.

- WP in G: given $w \in F(X)$ decide if $w = 1$ in G.
- CP in G: given two words $u, v \in F(X)$ decide if u and v are conjugate in G.
- MP in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ decide if $w \in H$.
- IP in a given class $\mathcal{C}$ of presentations: given two finite (or recursive) presentations in $\mathcal{C}$ decide if the groups, defined by the presentations, are isomorphic or not.

Decision algorithms

In general, an algorithmic problem consists of a set I of inputs or instances of the problem and a question whether or not an input $w \in I$ satisfies a given property $P(w)$, i.e., it is a pair (I, P).

A problem (I, P) is decidable if **there exists** an algorithm (decision algorithm) $\mathcal{A}$ that for a given $w \in I$ decides if $P(w)$ holds or not.

Notice, it is not required here to find a decision algorithm $\mathcal{A}$!?

Decision algorithms

In general, an algorithmic problem consists of a set I of inputs or instances of the problem and a question whether or not an input $w \in I$ satisfies a given property $P(w)$, i.e., it is a pair (I, P).

A problem (I, P) is decidable if **there exists** an algorithm (decision algorithm) $\mathcal{A}$ that for a given $w \in I$ decides if $P(w)$ holds or not.

Notice, it is not required here to find a decision algorithm $\mathcal{A}$!?

Decision algorithms

In general, an algorithmic problem consists of a set I of inputs or instances of the problem and a question whether or not an input $w \in I$ satisfies a given property $P(w)$, i.e., it is a pair (I, P).

A problem (I, P) is decidable if **there exists** an algorithm (decision algorithm) $\mathcal{A}$ that for a given $w \in I$ decides if $P(w)$ holds or not.

Notice, it is not required here to find a decision algorithm $\mathcal{A}$!?

Unsettling times

Turing:

The Halting Problem for Turing machines is undecidable.

Still, common viewpoint: all "real" problems are decidable.

Novikov (1955):

There exists a finitely presented group G with undecidable word problem.
To prove these one really needs to have a rigorous notion of a computable function (Turing, Markov, Post).

Unsettling times

Turing:

The Halting Problem for Turing machines is undecidable.

Still, common viewpoint: all "real" problems are decidable.

Novikov (1955):

There exists a finitely presented group G with undecidable word problem.
To prove these one really needs to have a rigorous notion of a computable function (Turing, Markov, Post).

Unsettling times

Turing:

The Halting Problem for Turing machines is undecidable.

Still, common viewpoint: all "real" problems are decidable.

Novikov (1955):

There exists a finitely presented group G with undecidable word problem.
To prove these one really needs to have a rigorous notion of a computable function (Turing, Markov, Post).

Decidable/undecidable

Main Theme: classification of decidable/undecidable problems.

Research splits:

Monsters hunting (find undecidable monsters wherever you can)

Positive thinking (find good decision algorithms for "small" classes)

Outcome (in group theory):

a lot of groups with undecidable word and conjugacy problems.

A lot of particular decision algorithms.

Common belief: Undecidable problems are very very hard!

Decidable/undecidable

Main Theme: classification of decidable/undecidable problems.

Research splits:

Monsters hunting (find undecidable monsters wherever you can)

Positive thinking (find good decision algorithms for "small" classes)

Outcome (in group theory):

a lot of groups with undecidable word and conjugacy problems.

A lot of particular decision algorithms.

Common belief: Undecidable problems are very very hard!

Decidable/undecidable

Main Theme: classification of decidable/undecidable problems.

Research splits:

Monsters hunting (find undecidable monsters wherever you can)

Positive thinking (find good decision algorithms for "small" classes)

Outcome (in group theory):

a lot of groups with undecidable word and conjugacy problems.

A lot of particular decision algorithms.

Common belief: Undecidable problems are very very hard!

New wave: getting practical

Time complexity.

Let $\mathcal{A}$ be an algorithm with inputs from a set S,

$|w|$ = the "size" of $w \in S$.

$T_{\mathcal{A}}(w)$ = the number of steps required for $\mathcal{A}$ to stop on the input $w \in S$.

$\mathcal{A}$ is in **polynomial time** if for some polynomial $p(x)$

$$T_{\mathcal{A}}(w) \leq p(|w|)$$

Main thesis: "Polynomial time = tractable"

Time complexity.

Let $\mathcal{A}$ be an algorithm with inputs from a set S,

$|w|$ = the "size" of $w \in S$.

$T_{\mathcal{A}}(w)$ = the number of steps required for $\mathcal{A}$ to stop on the input $w \in S$.

$\mathcal{A}$ is in **polynomial time** if for some polynomial $p(x)$

$$T_{\mathcal{A}}(w) \leq p(|w|)$$

Main thesis: "Polynomial time = tractable"

Time complexity.

Let $\mathcal{A}$ be an algorithm with inputs from a set S,

$|w|$ = the "size" of $w \in S$.

$T_{\mathcal{A}}(w)$ = the number of steps required for $\mathcal{A}$ to stop on the input $w \in S$.

$\mathcal{A}$ is in **polynomial time** if for some polynomial $p(x)$

$$T_{\mathcal{A}}(w) \leq p(|w|)$$

Main thesis: "Polynomial time = tractable"

Complexity

New wave main themes:

- Estimate the time complexity of decidable algorithmic problems.
- Estimate the space complexity of decidable algorithmic problems.
- Estimate other resources required to solve algorithmic problems.

Complexity

New wave main themes:

- Estimate the time complexity of decidable algorithmic problems.
- Estimate the space complexity of decidable algorithmic problems.
- Estimate other resources required to solve algorithmic problems.

Complexity

New wave main themes:

- Estimate the time complexity of decidable algorithmic problems.
- Estimate the space complexity of decidable algorithmic problems.
- Estimate other resources required to solve algorithmic problems.

NP-hard problems

Cook and Levin:

NP = polynomial time by non-deterministic Turing machines.

Many real problems are in **NP**.

NP-complete problems = the hardest in **NP**

There are many NP-complete problems known: 3-SAT, TSP, Hamilton cycle,...

Common belief: NP-complete problems are very very hard!

NP-hard problems

Cook and Levin:

NP = polynomial time by non-deterministic Turing machines.

Many real problems are in **NP**.

NP-complete problems = the hardest in **NP**

There are many NP-complete problems known: 3-SAT, TSP, Hamilton cycle,...

Common belief: NP-complete problems are very very hard!

Modern times: getting really practical

Main theme:

- Take tractable decision problems.
- Implement the (fast) decision algorithms.
- Design software packages and use them in practice (group theoretic research or cryptanalysis).

Modern times: getting really practical

Main theme:

- Take tractable decision problems.
- Implement the (fast) decision algorithms.
- Design software packages and use them in practice (group theoretic research or cryptanalysis).

Modern times: getting really practical

Main theme:

- Take tractable decision problems.
- Implement the (fast) decision algorithms.
- Design software packages and use them in practice (group theoretic research or cryptanalysis).

Problems, problems, problems

Somehow, it does not really work...

The whole theory is made for theoretical use only,

it just does not fit practical computations,

so, mostly, it does not work as it seems it should be.

For example, everyone, absolutely everyone, knows that WP in a given free group is awfully easy. Or it seems easy until you think about an algorithm to be implemented in a software package.

Problems, problems, problems

Somehow, it does not really work...

The whole theory is made for theoretical use only,

it just does not fit practical computations,

so, mostly, it does not work as it seems it should be.

For example, everyone, absolutely everyone, knows that WP in a given free group is awfully easy. Or it seems easy until you think about an algorithm to be implemented in a software package.

Problems, problems, problems

Somehow, it does not really work...

The whole theory is made for theoretical use only,

it just does not fit practical computations,

so, mostly, it does not work as it seems it should be.

For example, everyone, absolutely everyone, knows that WP in a given free group is awfully easy. Or it seems easy until you think about an algorithm to be implemented in a software package.

Problems, problems, problems

Somehow, it does not really work...

The whole theory is made for theoretical use only,

it just does not fit practical computations,

so, mostly, it does not work as it seems it should be.

For example, everyone, absolutely everyone, knows that WP in a given free group is awfully easy. Or it seems easy until you think about an algorithm to be implemented in a software package.

Problems, problems, problems

Somehow, it does not really work...

The whole theory is made for theoretical use only,

it just does not fit practical computations,

so, mostly, it does not work as it seems it should be.

For example, everyone, absolutely everyone, knows that WP in a given free group is awfully easy. Or it seems easy until you think about an algorithm to be implemented in a software package.

Problems, problems, problems

Somehow, it does not really work...

The whole theory is made for theoretical use only,

it just does not fit practical computations,

so, mostly, it does not work as it seems it should be.

For example, everyone, absolutely everyone, knows that WP in a given free group is awfully easy. Or it seems easy until you think about an algorithm to be implemented in a software package.

Looking at the beginning again

Given $G = \langle X \mid R \rangle$.

WP in G is decidable if there exists and algorithm $\mathcal{A}$ that for a given $w \in F(X)$ decides if $w = 1$ in G.

Notice (exercise), that if WP in G is decidable with respect to one finite presentation then it is decidable in G for any finite presentation. Nice!.

Innocent task: Given $G = \langle X \mid R \rangle$ and $\mathcal{A}$, as above, find a decision algorithm $\mathcal{A}'$ for WP in G relative to another given presentation $G = \langle X' \mid R' \rangle$.

So, please, implement an algorithm, you want your package to be general enough.

Looking at the beginning again

Given $G = \langle X \mid R \rangle$.

WP in G is decidable if there exists and algorithm $\mathcal{A}$ that for a given $w \in F(X)$ decides if $w = 1$ in G.

Notice (exercise), that if WP in G is decidable with respect to one finite presentation then it is decidable in G for any finite presentation. Nice!.

Innocent task: Given $G = \langle X \mid R \rangle$ and $\mathcal{A}$, as above, find a decision algorithm $\mathcal{A}'$ for WP in G relative to another given presentation $G = \langle X' \mid R' \rangle$.

So, please, implement an algorithm, you want your package to be general enough.

Looking at the beginning again

Given $G = \langle X \mid R \rangle$.

WP in G is decidable if there exists and algorithm $\mathcal{A}$ that for a given $w \in F(X)$ decides if $w = 1$ in G.

Notice (exercise), that if WP in G is decidable with respect to one finite presentation then it is decidable in G for any finite presentation. Nice!.

Innocent task: Given $G = \langle X \mid R \rangle$ and $\mathcal{A}$, as above, find a decision algorithm $\mathcal{A}'$ for WP in G relative to another given presentation $G = \langle X' \mid R' \rangle$.

So, please, implement an algorithm, you want your package to be general enough.

Looking at the beginning again

Given $G = \langle X \mid R \rangle$.

WP in G is decidable if there exists and algorithm $\mathcal{A}$ that for a given $w \in F(X)$ decides if $w = 1$ in G.

Notice (exercise), that if WP in G is decidable with respect to one finite presentation then it is decidable in G for any finite presentation. Nice!.

Innocent task: Given $G = \langle X \mid R \rangle$ and $\mathcal{A}$, as above, find a decision algorithm $\mathcal{A}'$ for WP in G relative to another given presentation $G = \langle X' \mid R' \rangle$.

So, please, implement an algorithm, you want your package to be general enough.

An algorithm

Do the following:

- Start with $\langle X \mid R \rangle$ and apply elementary Tietze transformations one-by one until $\langle X' \mid R' \rangle$ occurs.
- Get an isomorphism of the groups defined by these presentations and rewrite generators in X' as words in X.
- Rewrite a given group word w' in the alphabet X' as a word W in X.
- Apply $\mathcal{A}$ to w.

Denote this algorithm by $\mathcal{B}$. Note, it depends on $\mathcal{A}$.

Of course, could be other algorithms.

An algorithm

Do the following:

- Start with $\langle X \mid R \rangle$ and apply elementary Tietze transformations one-by one until $\langle X' \mid R' \rangle$ occurs.
- Get an isomorphism of the groups defined by these presentations and rewrite generators in X' as words in X.
- Rewrite a given group word w' in the alphabet X' as a word W in X.
- Apply $\mathcal{A}$ to w.

Denote this algorithm by $\mathcal{B}$. Note, it depends on $\mathcal{A}$.

Of course, could be other algorithms.

Problem

Question: Is $\mathcal{B}$ tractable?

Answer: Never.

Reason: we used total enumeration of Tietze transformations and the related search for an isomorphism of groups given by presentations $\langle X \mid R \rangle$ and $\langle X' \mid R' \rangle$ - it is provably non-tractable.

Problem

Question: Is $\mathcal{B}$ tractable?

Answer: Never.

Reason: we used total enumeration of Tietze transformations and the related search for an isomorphism of groups given by presentations $\langle X \mid R \rangle$ and $\langle X' \mid R' \rangle$ - it is provably non-tractable.

Problem

Question: Is $\mathcal{B}$ tractable?

Answer: Never.

Reason: we used total enumeration of Tietze transformations and the related search for an isomorphism of groups given by presentations $\langle X \mid R \rangle$ and $\langle X' \mid R' \rangle$ - it is provably non-tractable.

Decision and Search

Nowadays, it is recognized that there are decision and search variations of algorithmic problems (I, P).

- The Search Word Problem (WP) in G: given $w \in F(X)t$, such that $w = 1$ in G, find a decomposition $w = \prod_{i=1}^{k} s_i^{-1} r_i s_i$, where $s_i \in F(X), r_i \in R$.
- The Search Conjugacy Problem (CP) in G: given two words $u, v \in F(X)$, which define conjugated elements in Gind a conjugator.
- The Search Membership Problem (MP) in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ which belongs to H, find its decomposition as a product of the generators of H.
- The Search Isomorphism problem (IP) in a given class $\mathcal{C}$ of presentations: given two presentations in $\mathcal{C}$ of isomorphic groups find an isomorphism.

Decision and Search

Nowadays, it is recognized that there are decision and search variations of algorithmic problems (I, P).

- The Search Word Problem (WP) in G: given $w \in F(X)t$, such that $w = 1$ in G, find a decomposition $w = \prod_{i=1}^{k} s_i^{-1} r_i s_i$, where $s_i \in F(X), r_i \in R$.
- The Search Conjugacy Problem (CP) in G: given two words $u, v \in F(X)$, which define conjugated elements in Gfind a conjugator.
- The Search Membership Problem (MP) in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ which belongs to H, find its decomposition as a product of the generators of H.
- The Search Isomorphism problem (IP) in a given class $\mathcal{C}$ of presentations: given two presentations in $\mathcal{C}$ of isomorphic groups find an isomorphism.

Decision and Search

Nowadays, it is recognized that there are decision and search variations of algorithmic problems (I, P).

- The Search Word Problem (WP) in G: given $w \in F(X)t$, such that $w = 1$ in G, find a decomposition $w = \prod_{i=1}^{k} s_i^{-1} r_i s_i$, where $s_i \in F(X), r_i \in R$.
- The Search Conjugacy Problem (CP) in G: given two words $u, v \in F(X)$, which define conjugated elements in Gind a conjugator.
- The Search Membership Problem (MP) in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ which belongs to H, find its decomposition as a product of the generators of H.
- The Search Isomorphism problem (IP) in a given class $\mathcal{C}$ of presentations: given two presentations in $\mathcal{C}$ of isomorphic groups find an isomorphism.

Decision and Search

Nowadays, it is recognized that there are decision and search variations of algorithmic problems (I, P).

- The Search Word Problem (WP) in G: given $w \in F(X)t$, such that $w = 1$ in G, find a decomposition $w = \prod_{i=1}^{k} s_i^{-1} r_i s_i$, where $s_i \in F(X), r_i \in R$.
- The Search Conjugacy Problem (CP) in G: given two words $u, v \in F(X)$, which define conjugated elements in Gfind a conjugator.
- The Search Membership Problem (MP) in G for a fixed subgroup $H \leq G$: given $w \in F(X)$ which belongs to H, find its decomposition as a product of the generators of H.
- The Search Isomorphism problem (IP) in a given class $\mathcal{C}$ of presentations: given two presentations in $\mathcal{C}$ of isomorphic groups find an isomorphism.

Finitely presented groups

Let $G = \langle X \mid R \rangle$ be finitely presented group.

What algorithmic advantages a finite presentation gives one?

Fact

The standard search algorithmic problems for G are decidable.

There are simple enumeration search algorithms.

Search algorithms

- Search WP: enumerate all products of the type $\prod_i^{-1} r_i s_i$, where $s_i \in F(X), r_i \in R^{\pm 1}$, into a sequence

$$w_1, w_2, \ldots, w_n, \ldots.$$

Given a word $w \in F(X)$ check one by one if the freely reduced form of w is equal the one of $w_1, w_2, \ldots$. If $w = 1$ in G eventually one will find $w_n = w$.

- Search IP: enumerate all finite sequences of Tietze transformations

$$\tau_1, \tau_2, \ldots, \tau_n \ldots,$$

Given two finite presentations P_1 and P_2 apply $\tau_1, \tau_2, \ldots$ to P_1 until obtaining P_2. By Tietze theorem if the groups defined by P_1 and P_2 are isomorphic then P_2 will eventually appear.

Search algorithms

- Search WP: enumerate all products of the type $\prod_i^{-1} r_i s_i$, where $s_i \in F(X), r_i \in R^{\pm 1}$, into a sequence

$$w_1, w_2, \ldots, w_n, \ldots.$$

Given a word $w \in F(X)$ check one by one if the freely reduced form of w is equal the one of $w_1, w_2, \ldots$. If $w = 1$ in G eventually one will find $w_n = w$.

- Search IP: enumerate all finite sequences of Tietze transformations

$$\tau_1, \tau_2, \ldots, \tau_n \ldots,$$

Given two finite presentations P_1 and P_2 apply $\tau_1, \tau_2, \ldots$ to P_1 until obtaining P_2. By Tietze theorem if the groups defined by P_1 and P_2 are isomorphic then P_2 will eventually appear.

Complexity of the general enumeration algorithms

Most of the general enumeration algorithms are absolutely impractical.

Theorem

If a decision problem is undecidable then there is no any recursive upper bound on the time complexity of the search variation of this problem.

For example, the decision IP is undecidable in the class of all finite presentations, so the search algorithm above is as inefficient as anything.

Complexity of the general enumeration algorithms

Most of the general enumeration algorithms are absolutely impractical.

Theorem

If a decision problem is undecidable then there is no any recursive upper bound on the time complexity of the search variation of this problem.

For example, the decision IP is undecidable in the class of all finite presentations, so the search algorithm above is as inefficient as anything.

Particular and uniform

There are particular and uniform variations of algorithmic problems.

The uniform variation requires to find an algorithm that works in a whole class of groups.

For example, computer programs are supposed to be applicable to a range of problems, not only to a particular instance of a program.

Of course, it is harder to find uniform decision algorithms.

Particular and uniform

- Solve WP or CP in the class of hyperbolic groups. Here an instance of the problem consists of a finite presentation $\langle X \mid R \rangle$ of a hyperbolic group G and a word $w \in F(X)$, and the question is whether $w = 1$ in G or not.
- Solve WP or CP in the class of one-relator groups.
- Solve WP or CP in the class of all metabelian or nilpotent groups.
- Give a decision (uniform) algorithm for MP for all finitely generated subgroups in G.

For example, Grobner-Shirshov algorithms are uniform MP algorithms. WP in G is a particular MP problem for the trivial subgroup 1.

Particular and uniform

- Solve WP or CP in the class of hyperbolic groups. Here an instance of the problem consists of a finite presentation $\langle X \mid R \rangle$ of a hyperbolic group G and a word $w \in F(X)$, and the question is whether $w = 1$ in G or not.
- Solve WP or CP in the class of one-relator groups.
- Solve WP or CP in the class of all metabelian or nilpotent groups.
- Give a decision (uniform) algorithm for MP for all finitely generated subgroups in G.

For example, Grobner-Shirshov algorithms are uniform MP algorithms. WP in G is a particular MP problem for the trivial subgroup 1.

Particular and uniform

- Solve WP or CP in the class of hyperbolic groups. Here an instance of the problem consists of a finite presentation $\langle X \mid R \rangle$ of a hyperbolic group G and a word $w \in F(X)$, and the question is whether $w = 1$ in G or not.
- Solve WP or CP in the class of one-relator groups.
- Solve WP or CP in the class of all metabelian or nilpotent groups.
- Give a decision (uniform) algorithm for MP for all finitely generated subgroups in G.

For example, Grobner-Shirshov algorithms are uniform MP algorithms. WP in G is a particular MP problem for the trivial subgroup 1.

Particular and uniform

- Solve WP or CP in the class of hyperbolic groups. Here an instance of the problem consists of a finite presentation $\langle X \mid R \rangle$ of a hyperbolic group G and a word $w \in F(X)$, and the question is whether $w = 1$ in G or not.
- Solve WP or CP in the class of one-relator groups.
- Solve WP or CP in the class of all metabelian or nilpotent groups.
- Give a decision (uniform) algorithm for MP for all finitely generated subgroups in G.

For example, Grobner-Shirshov algorithms are uniform MP algorithms. WP in G is a particular MP problem for the trivial subgroup 1.

One relator groups

The uniform WP is decidable in the class of one relator groups (Magnus, 1930s).

Complexity of the Magnus breakdown process is unknown (it is believed to be hard).

Conjecture

WP in each given one-relator groups is Ptime decidable.

WP is linear time in "most" one relator groups (from Gromov).

One relator groups

The uniform WP is decidable in the class of one relator groups (Magnus, 1930s).

Complexity of the Magnus breakdown process is unknown (it is believed to be hard).

Conjecture

WP in each given one-relator groups is Ptime decidable.

WP is linear time in "most" one relator groups (from Gromov).

One relator groups

The uniform WP is decidable in the class of one relator groups (Magnus, 1930s).

Complexity of the Magnus breakdown process is unknown (it is believed to be hard).

Conjecture

WP in each given one-relator groups is Ptime decidable.

WP is linear time in "most" one relator groups (from Gromov).

One relator groups

The uniform WP is decidable in the class of one relator groups (Magnus, 1930s).

Complexity of the Magnus breakdown process is unknown (it is believed to be hard).

Conjecture

WP in each given one-relator groups is Ptime decidable.

WP is linear time in "most" one relator groups (from Gromov).

One relator groups

The most difficult known one relator group is the so-called Baumslag-Gersten group

$$G = \left\langle a, b \ ; \ a^{a^b} = a^2 \right\rangle.$$

Theorem, Gersten

The Dehn function of G is not bounded by any finite tower of exponents.

Theorem, Myasnikov, Ushakov

The WP in G is polynomial time decidable.

Based on the work of Kapovich and Schupp, who proved that WP in G is at most exponential. We used new type of arithmetic circuits (decorated graphs).

One relator groups

The most difficult known one relator group is the so-called Baumslag-Gersten group

$$G = \left\langle a, b \ ; \ a^{a^b} = a^2 \right\rangle.$$

Theorem, Gersten

The Dehn function of G is not bounded by any finite tower of exponents.

Theorem, Myasnikov, Ushakov

The WP in G is polynomial time decidable.

Based on the work of Kapovich and Schupp, who proved that WP in G is at most exponential. We used new type of arithmetic circuits (decorated graphs).

One relator groups

The most difficult known one relator group is the so-called Baumslag-Gersten group

$$G = \left\langle a, b \ ; \ a^{a^b} = a^2 \right\rangle.$$

Theorem, Gersten

The Dehn function of G is not bounded by any finite tower of exponents.

Theorem, Myasnikov, Ushakov

The WP in G is polynomial time decidable.

Based on the work of Kapovich and Schupp, who proved that WP in G is at most exponential. We used new type of arithmetic circuits (decorated graphs).

One relator groups

The most difficult known one relator group is the so-called Baumslag-Gersten group

$$G = \left\langle a, b \ ; \ a^{a^b} = a^2 \right\rangle.$$

Theorem, Gersten

The Dehn function of G is not bounded by any finite tower of exponents.

Theorem, Myasnikov, Ushakov

The WP in G is polynomial time decidable.

Based on the work of Kapovich and Schupp, who proved that WP in G is at most exponential. We used new type of arithmetic circuits (decorated graphs).

Hyperbolic groups

Again, everybody knows that WP in a given hyperbolic groups is linear time! It can be done by the famous Dehn's algorithm (very simple length reducing rewriting system).

Try to program, good luck.

The point is that the Dehn's algorithm works in a hyperbolic $G = \langle X \mid R \rangle$ if and only if the presentation $\langle X \mid R \rangle$ is Dehn.

Hence one needs to find a Dehn presentation of G, starting with an arbitrary presentation $G = \langle X \mid R \rangle$.

Hyperbolic groups

Again, everybody knows that WP in a given hyperbolic groups is linear time! It can be done by the famous Dehn's algorithm (very simple length reducing rewriting system).

Try to program, good luck.

The point is that the Dehn's algorithm works in a hyperbolic $G = \langle X \mid R \rangle$ if and only if the presentation $\langle X \mid R \rangle$ is Dehn.

Hence one needs to find a Dehn presentation of G, starting with an arbitrary presentation $G = \langle X \mid R \rangle$.

Hyperbolic groups

Again, everybody knows that WP in a given hyperbolic groups is linear time! It can be done by the famous Dehn's algorithm (very simple length reducing rewriting system).

Try to program, good luck.

The point is that the Dehn's algorithm works in a hyperbolic $G = \langle X \mid R \rangle$ if and only if the presentation $\langle X \mid R \rangle$ is Dehn.

Hence one needs to find a Dehn presentation of G, starting with an arbitrary presentation $G = \langle X \mid R \rangle$.

Dehn presentations

To find a Dehn presentation for $G = \langle X \mid R \rangle$ do the following:

- Find a hyperbolicity constant δ for G (it is possible, due to Papasoglu and Holt),
- add to R all relation of length up to 8δ (seems like exponential already).

Question: Is there a tractable algorithm to find δ?

Answer: No.

The property of being hyperbolic is a Markov property, so it is undecidable if a given presentation defines a hyperbolic group. Hence the search problem of finding δ cannot be bounded by a recursive function.

Dehn presentations

To find a Dehn presentation for $G = \langle X \mid R \rangle$ do the following:

- Find a hyperbolicity constant δ for G (it is possible, due to Papasoglu and Holt),
- add to R all relation of length up to 8δ (seems like exponential already).

Question: Is there a tractable algorithm to find δ?

Answer: No.

The property of being hyperbolic is a Markov property, so it is undecidable if a given presentation defines a hyperbolic group. Hence the search problem of finding δ cannot be bounded by a recursive function.

Dehn presentations

To find a Dehn presentation for $G = \langle X \mid R \rangle$ do the following:

- Find a hyperbolicity constant δ for G (it is possible, due to Papasoglu and Holt),
- add to R all relation of length up to 8δ (seems like exponential already).

Question: Is there a tractable algorithm to find δ?

Answer: No.

The property of being hyperbolic is a Markov property, so it is undecidable if a given presentation defines a hyperbolic group. Hence the search problem of finding δ cannot be bounded by a recursive function.

Uniform Search MP in free groups

Question: Is the Uniform Search Membership Problem in a given free group F in P?

Very deep question related to the uniform polynomial bounds on the size of inverse automorphisms of F.

Uniform Search MP in free groups

Question: Is the Uniform Search Membership Problem in a given free group F in P?

Very deep question related to the uniform polynomial bounds on the size of inverse automorphisms of F.

Tractable problems: another look

Problems in **P** are assumed to be **tractable** (doible by real computers).

Are there any other tractable problems?

Recall that the classical time complexity is complexity on the *worst possible inputs*, - hence the modern name - **worst case complexity**.

Observe, that these worst-case inputs could be *very very sparse* - may not ever appear in real computations!

Hence the problem still might be tractable.

Tractable problems: another look

Problems in **P** are assumed to be **tractable** (doible by real computers).

Are there any other tractable problems?

Recall that the classical time complexity is complexity on the *worst possible inputs*, - hence the modern name - **worst case complexity**.

Observe, that these worst-case inputs could be *very very sparse* - may not ever appear in real computations!

Hence the problem still might be tractable.

Tractable problems: another look

Problems in **P** are assumed to be **tractable** (doible by real computers).

Are there any other tractable problems?

Recall that the classical time complexity is complexity on the *worst possible inputs*, - hence the modern name - **worst case complexity**.

Observe, that these worst-case inputs could be *very very sparse* - may not ever appear in real computations!

Hence the problem still might be tractable.

Average case complexity

Let μ be a measure on the set of inputs S of the algorithm $\mathcal{A}$.

Expected running time of the algorithm $\mathcal{A}$ on the set S:

$$\int_S T_{\mathcal{A}}(w)\mu(w)$$

Levin (1986), Gurevich (1987):

- Introduced average P and NP.
- Average case NP-complete problems.

Common belief: Average case NP-complete problems are very very hard!

Average case complexity

Let μ be a measure on the set of inputs S of the algorithm $\mathcal{A}$.

Expected running time of the algorithm $\mathcal{A}$ on the set S:

$$\int_S T_{\mathcal{A}}(w)\mu(w)$$

Levin (1986), Gurevich (1987):

- Introduced average P and NP.
- Average case NP-complete problems.

Common belief: Average case NP-complete problems are very very hard!

Average Case vs Worst Case:

Gurevich - Shelah (1987):

To find a Hamilton cycle (a closed path that contains every vertex exactly once) in a finite graph is linear time on average.

Some NP-complete problems are linear on average!

NP-completeness is not a proper measure for hardness of algorithmic problems (despite a common belief)

Recall the Simplex Algorithm for linear programming. This algorithm is of exponential time (worst case) but nevertheless it is used hundreds of times daily and in practice almost always works quickly.

Average Case vs Worst Case:

Gurevich - Shelah (1987):

To find a Hamilton cycle (a closed path that contains every vertex exactly once) in a finite graph is linear time on average.

Some NP-complete problems are linear on average!

NP-completeness is not a proper measure for hardness of algorithmic problems (despite a common belief)

Recall the Simplex Algorithm for linear programming. This algorithm is of exponential time (worst case) but nevertheless it is used hundreds of times daily and in practice almost always works quickly.

Average Case vs Worst Case:

Gurevich - Shelah (1987):

To find a Hamilton cycle (a closed path that contains every vertex exactly once) in a finite graph is linear time on average.

Some NP-complete problems are linear on average!

NP-completeness is not a proper measure for hardness of algorithmic problems (despite a common belief)

Recall the Simplex Algorithm for linear programming. This algorithm is of exponential time (worst case) but nevertheless it is used hundreds of times daily and in practice almost always works quickly.

Generic complexity

Generic complexity = complexity on most inputs.
Let μ be a (pseudo) measure on a set S.
A subset $Q \subset S$ is *generic* if $\mu(Q) = 1$, and *negligible* if $\mu(Q) = 0$.

A polynomial $f(n)$ is a *generic upper bound* for an algorithm $\mathcal{A}$ if there exists a *generic set* $Q \subset S$ such that for each $w \in Q$

$$T_{\mathcal{A}}(w) \leq f(|w|)$$

Complexity of $\mathcal{A}$

Time complexity of $\mathcal{A}$ is the time (number of steps) required for $\mathcal{A}$ to produce an answer on an input w relative to the size of w.

The space of inputs I comes equipped with a size function $s : I \to \mathbb{N}$: the length of a word , the size of a program representing w, etc.

The time function of $\mathcal{A}$:

$T_{\mathcal{A}}(n) =$ the maximal time spent by $\mathcal{A}$ on an input of size n.

$\mathcal{A}$ is in Ptime (linear, quadratic, etc) if $T_{\mathcal{A}}(n) = O(n^k)$ for some k.

Complexity of $\mathcal{A}$

Time complexity of $\mathcal{A}$ is the time (number of steps) required for $\mathcal{A}$ to produce an answer on an input w relative to the size of w.

The space of inputs I comes equipped with a size function $s : I \to \mathbb{N}$: the length of a word , the size of a program representing w, etc.

The time function of $\mathcal{A}$:

$T_{\mathcal{A}}(n) =$ the maximal time spent by $\mathcal{A}$ on an input of size n.

$\mathcal{A}$ is in Ptime (linear, quadratic, etc) if $T_{\mathcal{A}}(n) = O(n^k)$ for some k.

Complexity of $\mathcal{A}$

Time complexity of $\mathcal{A}$ is the time (number of steps) required for $\mathcal{A}$ to produce an answer on an input w relative to the size of w.

The space of inputs I comes equipped with a size function $s : I \to \mathbb{N}$: the length of a word , the size of a program representing w, etc.

The time function of $\mathcal{A}$:

$T_{\mathcal{A}}(n) =$ the maximal time spent by $\mathcal{A}$ on an input of size n.

$\mathcal{A}$ is in Ptime (linear, quadratic, etc) if $T_{\mathcal{A}}(n) = O(n^k)$ for some k.

Complexities

There are several types of complexity:

- Worst case (defined above): measuring hardness on the worst possible inputs. This is the classical type.
- Average case: gives the expected time (averaging over all inputs).
- Generic case: complexity on the typical (generic) inputs. Useful in applications, in particular, in cryptography.

Generic complexity is a very active area of research in algorithmic group theory. But I will not say much about it.

Complexities

There are several types of complexity:

- Worst case (defined above): measuring hardness on the worst possible inputs. This is the classical type.
- Average case: gives the expected time (averaging over all inputs).
- Generic case: complexity on the typical (generic) inputs. Useful in applications, in particular, in cryptography.

Generic complexity is a very active area of research in algorithmic group theory. But I will not say much about it.

Complexities

There are several types of complexity:

- Worst case (defined above): measuring hardness on the worst possible inputs. This is the classical type.
- Average case: gives the expected time (averaging over all inputs).
- Generic case: complexity on the typical (generic) inputs. Useful in applications, in particular, in cryptography.

Generic complexity is a very active area of research in algorithmic group theory. But I will not say much about it.

Complexities

There are several types of complexity:

- Worst case (defined above): measuring hardness on the worst possible inputs. This is the classical type.
- Average case: gives the expected time (averaging over all inputs).
- Generic case: complexity on the typical (generic) inputs. Useful in applications, in particular, in cryptography.

Generic complexity is a very active area of research in algorithmic group theory. But I will not say much about it.

The first philosophical point

A large part of algorithmic group theory can be explained in terms of "perfect" rewriting systems on words, or graphs, or complexes (van Kampen diagrams), or some other objects.

In this case, one rewrites words or graphs (or some other objects) according to a system of reduction rules, until it is possible, always terminating in a finite number of steps (terminating systems).

The resulting reduced words (graphs, objects) are either unique (confluence) or can be obtained from one another by a sequence of simple transformations (pre-perfect).

The first philosophical point

A large part of algorithmic group theory can be explained in terms of "perfect" rewriting systems on words, or graphs, or complexes (van Kampen diagrams), or some other objects.

In this case, one rewrites words or graphs (or some other objects) according to a system of reduction rules, until it is possible, always terminating in a finite number of steps (terminating systems).

The resulting reduced words (graphs, objects) are either unique (confluence) or can be obtained from one another by a sequence of simple transformations (pre-perfect).

Rewriting algorithms

The following are some famous rewriting algorithms of this sort:

- Finite complete rewriting systems (a direct analog of Grobner-Shirshov rewriting algorithms).
- Nielsen Method in free groups.
- Hall collection process in nilpotent and polycyclic groups.
- Stallings' folding method for MP in free groups (graph rewriting).
- Whitehead method for automorphisms of free groups (deciding whether two elements of $F(X)$ are in the same automorphic orbit).
- Dehn's algorithm in hyperbolic groups.

Rewriting algorithms

The following are some famous rewriting algorithms of this sort:

- Finite complete rewriting systems (a direct analog of Grobner-Shirshov rewriting algorithms).
- Nielsen Method in free groups.
- Hall collection process in nilpotent and polycyclic groups.
- Stallings' folding method for MP in free groups (graph rewriting).
- Whitehead method for automorphisms of free groups (deciding whether two elements of $F(X)$ are in the same automorphic orbit).
- Dehn's algorithm in hyperbolic groups.

Rewriting algorithms

The following are some famous rewriting algorithms of this sort:

- Finite complete rewriting systems (a direct analog of Grobner-Shirshov rewriting algorithms).
- Nielsen Method in free groups.
- Hall collection process in nilpotent and polycyclic groups.
- Stallings' folding method for MP in free groups (graph rewriting).
- Whitehead method for automorphisms of free groups (deciding whether two elements of $F(X)$ are in the same automorphic orbit).
- Dehn's algorithm in hyperbolic groups.

Rewriting algorithms

The following are some famous rewriting algorithms of this sort:

- Finite complete rewriting systems (a direct analog of Grobner-Shirshov rewriting algorithms).
- Nielsen Method in free groups.
- Hall collection process in nilpotent and polycyclic groups.
- Stallings' folding method for MP in free groups (graph rewriting).
- Whitehead method for automorphisms of free groups (deciding whether two elements of $F(X)$ are in the same automorphic orbit).
- Dehn's algorithm in hyperbolic groups.

Rewriting algorithms

The following are some famous rewriting algorithms of this sort:

- Finite complete rewriting systems (a direct analog of Grobner-Shirshov rewriting algorithms).
- Nielsen Method in free groups.
- Hall collection process in nilpotent and polycyclic groups.
- Stallings' folding method for MP in free groups (graph rewriting).
- Whitehead method for automorphisms of free groups (deciding whether two elements of $F(X)$ are in the same automorphic orbit).
- Dehn's algorithm in hyperbolic groups.

Rewriting algorithms

The following are some famous rewriting algorithms of this sort:

- Finite complete rewriting systems (a direct analog of Grobner-Shirshov rewriting algorithms).
- Nielsen Method in free groups.
- Hall collection process in nilpotent and polycyclic groups.
- Stallings' folding method for MP in free groups (graph rewriting).
- Whitehead method for automorphisms of free groups (deciding whether two elements of $F(X)$ are in the same automorphic orbit).
- Dehn's algorithm in hyperbolic groups.

The second philosophical point

One can get a "perfect" rewriting system (perhaps, an infinite one) in the following way:

- start with a naive one, for example, given a presentation $\langle X \mid R \rangle$ rewrite relations as $\ell_i = r_i$, $|\ell_i| \geq |r_i|$, and form the associate system as a collection of rules $\ell_i \to r_i$.
- apply a kind of Knuth-Bendix (KB) completion process:
  - if there is an ambiguity (a critical pair) resolve it by adding a new rule,
  - clean up,
  - repeat,
- If the process halts the resulting systems is a finite perfect systems, otherwise, it it produces an infinite perfect system.

The second philosophical point

One can get a "perfect" rewriting system (perhaps, an infinite one) in the following way:

- start with a naive one, for example, given a presentation $\langle X \mid R \rangle$ rewrite relations as $\ell_i = r_i$, $|\ell_i| \geq |r_i|$, and form the associate system as a collection of rules $\ell_i \longrightarrow r_i$.
- apply a kind of Knuth-Bendix (KB) completion process:
  - if there is an ambiguity (a critical pair) resolve it by adding a new rule,
  - clean up,
  - repeat,
- If the process halts the resulting systems is a finite perfect systems, otherwise, it it produces an infinite perfect system.

The second philosophical point

One can get a "perfect" rewriting system (perhaps, an infinite one) in the following way:

- start with a naive one, for example, given a presentation $\langle X \mid R \rangle$ rewrite relations as $\ell_i = r_i$, $|\ell_i| \geq |r_i|$, and form the associate system as a collection of rules $\ell_i \longrightarrow r_i$.
- apply a kind of Knuth-Bendix (KB) completion process:
  - if there is an ambiguity (a critical pair) resolve it by adding a new rule,
  - clean up,
  - repeat,
- If the process halts the resulting systems is a finite perfect systems, otherwise, it it produces an infinite perfect system.

The second philosophical point

One can get a "perfect" rewriting system (perhaps, an infinite one) in the following way:

- start with a naive one, for example, given a presentation $\langle X \mid R \rangle$ rewrite relations as $\ell_i = r_i$, $|\ell_i| \geq |r_i|$, and form the associate system as a collection of rules $\ell_i \longrightarrow r_i$.
- apply a kind of Knuth-Bendix (KB) completion process:
  - if there is an ambiguity (a critical pair) resolve it by adding a new rule,
  - clean up,
  - repeat,
- If the process halts the resulting systems is a finite perfect systems, otherwise, it it produces an infinite perfect system.

The second philosophical point

One can get a "perfect" rewriting system (perhaps, an infinite one) in the following way:

- start with a naive one, for example, given a presentation $\langle X \mid R \rangle$ rewrite relations as $\ell_i = r_i$, $|\ell_i| \geq |r_i|$, and form the associate system as a collection of rules $\ell_i \longrightarrow r_i$.
- apply a kind of Knuth-Bendix (KB) completion process:
  - if there is an ambiguity (a critical pair) resolve it by adding a new rule,
  - clean up,
  - repeat,
- If the process halts the resulting systems is a finite perfect systems, otherwise, it it produces an infinite perfect system.

The third philosophical point

Finite does not mean better!

There are natural infinite perfect rewriting systems which are better then the finite ones.

Even in terms of computational complexity of the rewriting algorithms.

The third philosophical point

Finite does not mean better!

There are natural infinite perfect rewriting systems which are better then the finite ones.

Even in terms of computational complexity of the rewriting algorithms.

frame

On the other hand, there are very efficient algorithms for infinitely presented groups.

Two examples:

Free Solvable groups

Theorem, M., Romankov, Ushakov, Vershik, 2008

Let $S_{d,r}$ be a free solvable group of solvability class d, and rank r.

- The time complexity of the Word Problem in $S_{d,r}$ is $O(n^3 rd)$.
- The geodesic problem is NP-complete.

Theorem, Vassileva, 2008

Let $S_{d,r}$ be a free solvable group of solvability class d, and rank r. Then the Conjugacy Problem in $S_{d,r}$ is decidable in time $O(n^5 rd)$.

Free Solvable groups

Theorem, M., Romankov, Ushakov, Vershik, 2008

Let $S_{d,r}$ be a free solvable group of solvability class d, and rank r.

- The time complexity of the Word Problem in $S_{d,r}$ is $O(n^3 rd)$.
- The geodesic problem is NP-complete.

Theorem, Vassileva, 2008

Let $S_{d,r}$ be a free solvable group of solvability class d, and rank r. Then the Conjugacy Problem in $S_{d,r}$ is decidable in time $O(n^5 rd)$.

Free Solvable groups

Theorem, M., Romankov, Ushakov, Vershik, 2008

Let $S_{d,r}$ be a free solvable group of solvability class d, and rank r.

- The time complexity of the Word Problem in $S_{d,r}$ is $O(n^3 rd)$.
- The geodesic problem is NP-complete.

Theorem, Vassileva, 2008

Let $S_{d,r}$ be a free solvable group of solvability class d, and rank r. Then the Conjugacy Problem in $S_{d,r}$ is decidable in time $O(n^5 rd)$.

The Grigorchuk group

Theorem

Let Γ be the first Grigorchuk group. Then:

- (Grigorchuk) The time complexity of the Word Problem in Γ is $O(nlogn)$.
- (Lysenok, Myasnikov, Ushakov) The Conjugacy Problem is decidable in polynomial time $O(n^7)$.
- (Grigorchuk) The Membership Problem is decidable (the time complexity is unknown).

The Grigorchuk group

Theorem

Let Γ be the first Grigorchuk group. Then:

- (Grigorchuk) The time complexity of the Word Problem in Γ is $O(nlogn)$.
- (Lysenok, Myasnikov, Ushakov) The Conjugacy Problem is decidable in polynomial time $O(n^7)$.
- (Grigorchuk) The Membership Problem is decidable (the time complexity is unknown).

The Grigorchuk group

Theorem

Let Γ be the first Grigorchuk group. Then:

- (Grigorchuk) The time complexity of the Word Problem in Γ is $O(nlogn)$.
- (Lysenok, Myasnikov, Ushakov) The Conjugacy Problem is decidable in polynomial time $O(n^7)$.
- (Grigorchuk) The Membership Problem is decidable (the time complexity is unknown).

Data structures

Conclusion:

- it seems the complexity of algorithmic problems does not depend on finiteness of presentations.
- To get an efficient algorithm one needs to find a suitable data structure (a compressed word, a graph, a complex) to represent the objects.
- Such a data structure governs the rewriting algorithms.

Data structures

Conclusion:

- it seems the complexity of algorithmic problems does not depend on finiteness of presentations.
- To get an efficient algorithm one needs to find a suitable data structure (a compressed word, a graph, a complex) to represent the objects.
- Such a data structure governs the rewriting algorithms.

Data structures

Conclusion:

- it seems the complexity of algorithmic problems does not depend on finiteness of presentations.
- To get an efficient algorithm one needs to find a suitable data structure (a compressed word, a graph, a complex) to represent the objects.
- Such a data structure governs the rewriting algorithms.

Basics on rewriting in monoids

A *rewriting relation* $\Longrightarrow$ over a set X is an arbitrary binary relation on X.

$\overset{*}{\Longrightarrow}$ is the reflexive and transitive closure of $\Longrightarrow$.

$\overset{*}{\Longleftrightarrow}$ is the symmetric, reflexive, and transitive closure of $\Longrightarrow$.

Definitions

The relation $\Longrightarrow \subseteq X \times X$ is called:

i) *confluent*, if $y \overset{*}{\Longleftarrow} x \overset{*}{\Longrightarrow} z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

ii) *locally confluent*, if $y \Longleftarrow x \Longrightarrow z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

iii) $\Longrightarrow \subseteq X \times X$ is called *terminating* (or *Noetherian*), if every infinite chain

$$x_0 \overset{*}{\Longrightarrow} x_1 \overset{*}{\Longrightarrow} \cdots x_{i-1} \overset{*}{\Longrightarrow} x_i \overset{*}{\Longrightarrow} \cdots$$

becomes stationary.

A relation $\Longrightarrow \subseteq X \times X$ is called *convergent* (or *complete*) if it is or locally confluent and terminating.

Definitions

The relation $\Longrightarrow \subseteq X \times X$ is called:

i) *confluent*, if $y \overset{*}{\Longleftarrow} x \overset{*}{\Longrightarrow} z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

ii) *locally confluent*, if $y \Longleftarrow x \Longrightarrow z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

iii) $\Longrightarrow \subseteq X \times X$ is called *terminating* (or *Noetherian*), if every infinite chain

$$x_0 \overset{*}{\Longrightarrow} x_1 \overset{*}{\Longrightarrow} \cdots x_{i-1} \overset{*}{\Longrightarrow} x_i \overset{*}{\Longrightarrow} \cdots$$

becomes stationary.

A relation $\Longrightarrow \subseteq X \times X$ is called *convergent* (or *complete*) if it is or locally confluent and terminating.

Definitions

The relation $\Longrightarrow \subseteq X \times X$ is called:

i) *confluent*, if $y \overset{*}{\Longleftarrow} x \overset{*}{\Longrightarrow} z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

ii) *locally confluent*, if $y \Longleftarrow x \Longrightarrow z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

iii) $\Longrightarrow \subseteq X \times X$ is called *terminating* (or *Noetherian*), if every infinite chain

$$x_0 \overset{*}{\Longrightarrow} x_1 \overset{*}{\Longrightarrow} \cdots x_{i-1} \overset{*}{\Longrightarrow} x_i \overset{*}{\Longrightarrow} \cdots$$

becomes stationary.

A relation $\Longrightarrow \subseteq X \times X$ is called *convergent* (or *complete*) if it is or locally confluent and terminating.

Definitions

The relation $\Longrightarrow \subseteq X \times X$ is called:

i) *confluent*, if $y \overset{*}{\Longleftarrow} x \overset{*}{\Longrightarrow} z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

ii) *locally confluent*, if $y \Longleftarrow x \Longrightarrow z$ implies $y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$ for some w,

iii) $\Longrightarrow \subseteq X \times X$ is called *terminating* (or *Noetherian*), if every infinite chain

$$x_0 \overset{*}{\Longrightarrow} x_1 \overset{*}{\Longrightarrow} \cdots x_{i-1} \overset{*}{\Longrightarrow} x_i \overset{*}{\Longrightarrow} \cdots$$

becomes stationary.

A relation $\Longrightarrow \subseteq X \times X$ is called *convergent* (or *complete*) if it is or locally confluent and terminating.

Rewriting in monoids

Let M be a monoid.

A *rewriting system* over M is a subset $S \subseteq M \times M$.

M defines the rewriting relation

$$x \underset{S}{\Longrightarrow} y \text{ if and only if } x = p\ell q,\ y = prq \text{ for some } (\ell, r) \in S.$$

The relation $\overset{*}{\underset{S}{\Longleftrightarrow}} \subseteq M \times M$ is a congruence on M.

$M/\overset{*}{\underset{S}{\Longleftrightarrow}}$ (or simply M/S) is the quotient monoid.

Rewriting systems S and T over a monoid M are *equivalent* if $M_S = M_T$.

Rewriting in monoids

Let M be a monoid.

A *rewriting system* over M is a subset $S \subseteq M \times M$.

M defines the rewriting relation

$$x \underset{S}{\Longrightarrow} y \text{ if and only if } x = p\ell q,\ y = prq \text{ for some } (\ell, r) \in S.$$

The relation $\overset{*}{\underset{S}{\Longleftrightarrow}} \subseteq M \times M$ is a congruence on M.

$M/\overset{*}{\underset{S}{\Longleftrightarrow}}$ (or simply M/S) is the quotient monoid.

Rewriting systems S and T over a monoid M are *equivalent* if $M_S = M_T$.

Rewriting in monoids

Let M be a monoid.

A *rewriting system* over M is a subset $S \subseteq M \times M$.

M defines the rewriting relation

$$x \underset{S}{\Longrightarrow} y \text{ if and only if } x = p\ell q,\ y = prq \text{ for some } (\ell, r) \in S.$$

The relation $\overset{*}{\underset{S}{\Longleftrightarrow}} \subseteq M \times M$ is a congruence on M.

$M/\overset{*}{\underset{S}{\Longleftrightarrow}}$ (or simply M/S) is the quotient monoid.

Rewriting systems S and T over a monoid M are *equivalent* if $M_S = M_T$.

Rewriting in monoids

If $(\ell, r) \in S$ we write $\ell \longrightarrow r \in S$.

$\ell \longleftrightarrow r \in S$ if both (ℓ, r) and (r, ℓ) are in S.

A word w is *S-irreducible* if no left-hand side ℓ of S occurs in w as a factor.

Rewriting in monoids

If $(\ell, r) \in S$ we write $\ell \longrightarrow r \in S$.

$\ell \longleftrightarrow r \in S$ if both (ℓ, r) and (r, ℓ) are in S.

A word w is *S-irreducible* if no left-hand side ℓ of S occurs in w as a factor.

Complete systems

Let S be a convergent (complete) rewriting system. Then:

1) S is confluent.
2) Every $\overset{*}{\underset{S}{\Longleftrightarrow}}$ equivalence class in Γ^* contains a unique S-reduced word.
3) If S is finite then for a given word $w \in \Gamma^*$ one can effectively find its unique S-reduced form $red(w)$ (just by subsequently rewriting the word w until the result is S-reduced).

$red(w)$ is the normal form of w in M/S (relative to S).

Groups with complete presentations

There are many examples of groups that have finite convergent presentations:
finite groups, polycyclic group, free groups, some geometric groups.

Knuth-Bendix Procedure

The Knuth-Bendix procedure (KB) for general rewriting systems (and for groups by R.Gilman).

Let $\succ$ be a reduction well-ordering on a free monoid Γ^* with basis Γ and $S \subseteq \Gamma^* \times \Gamma^*$ a finite rewriting system compatible with $\succ$ ($\ell \succ r$ for $(\ell, r) \in S$).

If there exists a finite convergent rewriting system $T \subseteq \Gamma^* \times \Gamma^*$ compatible with $\succ$ and equivalent to S, then the Knuth-Bendix procedure finds one in finitely many steps.

No free lunch

- The time complexity of the word problem in a monoid M_S defined by a finite convergent system S may be of an arbitrarily high complexity [Otto].
- It may happen that WP in M_S is decidable in polynomial time, whereas the complexity of the standard rewriting algorithm that finds the S-reduced forms can be of an arbitrarily high complexity.
- The Knuth-Bendix procedure really depends on the chosen ordering $\succ$ [Otto].
- The problem of whether or not a given finitely presented group can be defined by a finite convergent rewriting system is undecidable [Ó'Dúnlaing].

No free lunch

- The time complexity of the word problem in a monoid M_S defined by a finite convergent system S may be of an arbitrarily high complexity [Otto].
- It may happen that WP in M_S is decidable in polynomial time, whereas the complexity of the standard rewriting algorithm that finds the S-reduced forms can be of an arbitrarily high complexity.
- The Knuth-Bendix procedure really depends on the chosen ordering $\succ$ [Otto].
- The problem of whether or not a given finitely presented group can be defined by a finite convergent rewriting system is undecidable [Ó'Dúnlaing].

No free lunch

- The time complexity of the word problem in a monoid M_S defined by a finite convergent system S may be of an arbitrarily high complexity [Otto].
- It may happen that WP in M_S is decidable in polynomial time, whereas the complexity of the standard rewriting algorithm that finds the S-reduced forms can be of an arbitrarily high complexity.
- The Knuth-Bendix procedure really depends on the chosen ordering $\succ$ [Otto].
- The problem of whether or not a given finitely presented group can be defined by a finite convergent rewriting system is undecidable [Ó'Dúnlaing].

Open problems

Problem

Is it true that every hyperbolic group has a finite convergent presentation?

It is known that some hyperbolic groups have finite convergent presentations, for example, surface groups [Chenadec].

Problem

Is it true that every finitely generated fully residually free group has a finite convergent presentation?

Open problems

Problem

Is it true that every hyperbolic group has a finite convergent presentation?

It is known that some hyperbolic groups have finite convergent presentations, for example, surface groups [Chenadec].

Problem

Is it true that every finitely generated fully residually free group has a finite convergent presentation?

Open problems

Problem

Do all automatic groups have finite convergent presentations?

Problem

Do all one-relator groups have finite convergent presentations?

Notice that all the groups above satisfy the homological condition FP_∞; which is the main known condition necessary for a group to have a finite convergent presentation [Squier].

Open problems

Problem

Do all automatic groups have finite convergent presentations?

Problem

Do all one-relator groups have finite convergent presentations?

Notice that all the groups above satisfy the homological condition FP_∞; which is the main known condition necessary for a group to have a finite convergent presentation [Squier].

Open problems

Problem

Do all automatic groups have finite convergent presentations?

Problem

Do all one-relator groups have finite convergent presentations?

Notice that all the groups above satisfy the homological condition FP_∞; which is the main known condition necessary for a group to have a finite convergent presentation [Squier].

Computing with infinite systems

An infinite string rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ can be used in computation provided:

- one has to be able to recognize if a given pair $(u, v) \in \Gamma^* \times \Gamma^*$ gives a rule $u \longrightarrow v \in S$ or not, i.e., the system S must be a recursive subset of $\Gamma^* \times \Gamma^*$.
- to rewrite with S one has to be able to check if for a given $u \in \Gamma^*$ there is a rule $\ell \longrightarrow r \in S$ with $\ell = u$, so we assume that the set $L(S)$ of the left-hand sides of the rules in S is a recursive subset of Γ^*. Systems satisfying these two conditions are termed *effective* rewriting systems.

Computing with infinite systems

An infinite string rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ can be used in computation provided:

- one has to be able to recognize if a given pair $(u, v) \in \Gamma^* \times \Gamma^*$ gives a rule $u \longrightarrow v \in S$ or not, i.e., the system S must be a recursive subset of $\Gamma^* \times \Gamma^*$.
- to rewrite with S one has to be able to check if for a given $u \in \Gamma^*$ there is a rule $\ell \longrightarrow r \in S$ with $\ell = u$, so we assume that the set $L(S)$ of the left-hand sides of the rules in S is a recursive subset of Γ^*. Systems satisfying these two conditions are termed *effective* rewriting systems.

Computing with infinite systems

For a recursive non-length-increasing system S one can effectively enumerate all the rules in S in such a way

$$\ell_0 \to r_0, \ell_1 \to r_1, \ldots, \ell_i \to r_i, \ldots \tag{1}$$

that if $i < j$ then $\ell_i \preceq \ell_j$ in the length-lexicographical ordering $\preceq$ and also if $\ell_i = \ell_j$ then $r_i \preceq r_j$. We call this enumeration of S *standard*.

Let S be an infinite effective convergent system. Then the word problem in the monoid M_S defined by S is decidable.

Computing with infinite systems

For a recursive non-length-increasing system S one can effectively enumerate all the rules in S in such a way

$$\ell_0 \to r_0, \ell_1 \to r_1, \ldots, \ell_i \to r_i, \ldots \tag{1}$$

that if $i < j$ then $\ell_i \preceq \ell_j$ in the length-lexicographical ordering $\preceq$ and also if $\ell_i = \ell_j$ then $r_i \preceq r_j$. We call this enumeration of S *standard*.

Let S be an infinite effective convergent system. Then the word problem in the monoid M_S defined by S is decidable.

Length-reducing and Dehn systems

S is *length-reducing* systems if $|\ell| > |r|$ for every rule $\ell \to r$ in S.

Lemma

If S is a finite length-reducing string rewriting system, then irreducible descendants of a given word can be computed in linear time (in the length of the word).

Dehn rewriting systems

Definition

A length-reducing string rewriting system which is confluent on the empty word is called a *Dehn* system.

If a group is defined by a finite length-reducing Dehn rewriting system then the rewriting algorithm is known in group theory as the *Dehn* algorithm.

Hyperbolic groups

It is known that, given a finite presentation of a hyperbolic group G, one can produce a finite Dehn presentation of G by adding, to a given presentation, all new relators of G up to some length (which depends on the hyperbolicity constant of G).

However, this algorithm is very inefficient and the following questions remain.

Problem

Is there a Knuth-Bendix type completion process that, given a finite presentation of a hyperbolic group G, finds a finite Dehn presentation of G.

Hyperbolic groups

It is known that, given a finite presentation of a hyperbolic group G, one can produce a finite Dehn presentation of G by adding, to a given presentation, all new relators of G up to some length (which depends on the hyperbolicity constant of G).

However, this algorithm is very inefficient and the following questions remain.

Problem

Is there a Knuth-Bendix type completion process that, given a finite presentation of a hyperbolic group G, finds a finite Dehn presentation of G.

Deciding Dehn presentations

Problem

Is there an algorithm that, given a finite presentation of a hyperbolic group, determines whether or not this presentation is Dehn.

Arzhantseva has shown that there is an algorithm that, given a finite presentation of a hyperbolic group and $\alpha \in [3/4, 1)$, detects whether or not this presentation is an α-Dehn presentation.

Here a presentation $\langle X \mid R \rangle$ of a group G is called an α-Dehn presentation if any non-empty freely reduced word $w \in (X \cup X^{-1})^*$ representing the identity in G contains as a factor a word u which is also a factor of a cyclic shift of some $r \in R^{\pm 1}$ with $|u| > \alpha |r|$.

Deciding Dehn presentations

Problem

Is there an algorithm that, given a finite presentation of a hyperbolic group, determines whether or not this presentation is Dehn.

Arzhantseva has shown that there is an algorithm that, given a finite presentation of a hyperbolic group and $\alpha \in [3/4, 1)$, detects whether or not this presentation is an α-Dehn presentation.

Here a presentation $\langle X \mid R \rangle$ of a group G is called an α-Dehn presentation if any non-empty freely reduced word $w \in (X \cup X^{-1})^*$ representing the identity in G contains as a factor a word u which is also a factor of a cyclic shift of some $r \in R^{\pm 1}$ with $|u| > \alpha |r|$.

Infinite length-reducing systems

Let $S \subseteq \Gamma^* \times \Gamma^*$ be an infinite recursive string rewriting system. Then the following hold.

1) If S is length-reducing then an irreducible descendant of a given word can be computed.
2) If S is Dehn and M_S is a group, then the word problem in M_S is decidable.

Infinite length-reducing systems

Let S be an effective non-length increasing rewriting system and

$$\ell_0 \to r_0, \ell_1 \to r_1, \ldots, \ell_i \to r_i, \ldots$$

its standard enumeration. If there an algorithm $\mathcal{A}$ and a polynomial $p(n)$ such that for every $n \in \mathbb{N}$ the algorithm $\mathcal{A}$ writes down the initial part of the standard enumeration of S with $|\ell_i| \leq n$ in time $p(n)$ then the system S is called *enumerable in time* $p(n)$ or *Ptime enumerable*.

Preperfect systems

Preperfect rewriting systems play an important part in solving the word problem and finding geodesics (shortest representatives in the equivalence classes) in groups.

Definition

A *Thue system* is a rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ such that the following conditions hold:

i) If $\ell \longrightarrow r \in S$ then $|\ell| \geq |r|$.

ii) If $\ell \longrightarrow r \in S$ with $|\ell| = |r|$, then $r \longrightarrow \ell \in S$, too.

Preperfect systems

Definition

A confluent Thue system is called *preperfect*.

If a rewriting system S is finite and preperfect, then one can decide the word problem in the monoid defined by S in polynomial space, and hence in exponential time. Moreover, along the way one can find an S-geodesic of a given word w, as well as, all S-geodesics of w.

Preperfect systems

Definition

A confluent Thue system is called *preperfect*.

If a rewriting system S is finite and preperfect, then one can decide the word problem in the monoid defined by S in polynomial space, and hence in exponential time. Moreover, along the way one can find an S-geodesic of a given word w, as well as, all S-geodesics of w.

KB for pre-perfect systems

In general an infinite number of *critical pairs* may appear in the Knuth-Bendix process, and one needs to be able to recognize when the current system becomes preperfect.

Unfortunately, this is algorithmically undecidable.

Theorem

The problem of verifying whether a finite Thue system is preperfect or not is undecidable.

KB for pre-perfect systems

In general an infinite number of *critical pairs* may appear in the Knuth-Bendix process, and one needs to be able to recognize when the current system becomes preperfect.

Unfortunately, this is algorithmically undecidable.

Theorem

The problem of verifying whether a finite Thue system is preperfect or not is undecidable.

Complexity

L

et S be an infinite preperfect rewriting system. Then:

1) if S is recursive then the word problem in the monoid M_S defined by S is decidable;
2) if S is Ptime enumerable then one can solve the word problem in M_S and find a geodesic of a given word in exponential time.

Geodesic systems

definition

A string rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ is called *geodesic* if S-geodesic words are exactly those words to which no length reducing rule from S can be applied.

Theorem

[GHHR] A group G is defined by a finite geodesic system S if and only if G is a finitely generated virtually free group.

Geodesic systems

definition

A string rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ is called *geodesic* if S-geodesic words are exactly those words to which no length reducing rule from S can be applied.

Theorem

[GHHR] A group G is defined by a finite geodesic system S if and only if G is a finitely generated virtually free group.

Decidability

Theorem

It is undecidable whether a finite rewriting system is geodesic.

Geodesically perfect systems

Definition

A string rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ is called *geodesically perfect*, if

i) S is geodesic, and

ii) if $u, v \in \Gamma^*$ are S-geodesics, then $u \overset{*}{\underset{S}{\Longleftrightarrow}} v$ if and only if $u \overset{*}{\underset{S_P}{\Longleftrightarrow}} v$, where S_P is the length-preserving part of S.

Knuth-Bendix completion for geodesically perfect systems

Theorem [DDM

There is a KB procedure for geodesically perfect systems.

Stallings' pregroups

A *pregroup* P is a set P with a distinguished element ε, equipped with a partial multiplication $m : D \to P$, $(a, b) \mapsto ab$, where $D \subseteq P \times P$, and an involution (or *inversion*), satisfying the following axioms for all $a, b, c, d \in P$.

(P1) $a\varepsilon$ and εa are defined and $a\varepsilon = \varepsilon a = a$;

(P2) $a^{-1}a$ and aa^{-1} are defined and $a^{-1}a = aa^{-1} = \varepsilon$;

(P3) if ab is defined, then so is $b^{-1}a^{-1}$, and $(ab)^{-1} = b^{-1}a^{-1}$;

(P4) if ab and bc are defined, then $(ab)c$ is defined if and only if $a(bc)$ is defined, in which case

$$(ab)c = a(bc);$$

(P5) if ab, bc, and cd are all defined then either abc or bcd is defined.

Stallings' pregroups and their universal groups

The *universal group* $U(P)$ of the pregroup P can be defined as the quotient monoid

$$U(P) = \Gamma^* / \{ ab = c \mid m(a,b) = c \},$$

where $\Gamma = P \setminus \{\varepsilon\}$ and $\varepsilon \in P$ is identified again with the empty word $1 \in \Gamma^*$.

Rewriting systems for pregroups

The system $S = S(P) \subseteq P^* \times P^*$
is defined by the following rules.

$$\begin{array}{rcll} \varepsilon & \longrightarrow & 1 & (= \text{the empty word}) \\ ab & \longrightarrow & [ab] & \text{if } (a,b) \in D \\ ab & \longleftrightarrow & [ac][c^{-1}b] & \text{if } (a,c), (c^{-1},b) \in D \end{array}$$

Theorem

Let P be a pregroup. Then the following hold.

1) $P^*/S(P) \simeq U(P)$.
2) S is a geodesically perfect Thue system.

Rewriting systems for pregroups

The system $S = S(P) \subseteq P^* \times P^*$
is defined by the following rules.

$$\begin{array}{rcll} \varepsilon & \longrightarrow & 1 & (= \text{the empty word}) \\ ab & \longrightarrow & [ab] & \text{if } (a,b) \in D \\ ab & \longleftrightarrow & [ac][c^{-1}b] & \text{if } (a,c), (c^{-1},b) \in D \end{array}$$

Theorem

Let P be a pregroup. Then the following hold.

1) $P^*/S(P) \simeq U(P)$.
2) S is a geodesically perfect Thue system.

Characterization of pregroups in terms of geodesic systems

We say that a rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ is *triangular* if each rule $\ell \to r \in S$ satisfies the "triangular" condition: $|\ell| = 2$, $|r| \leq 1$, so every rule in S is of the form $ab \to c$ where $a, b \in \Gamma$ and $c \in \Gamma \cup \{1\}$.

Theorem

Let P be a pregroup. Then the reduced part of the rewriting system $S'(P)$ is a geodesic triangular group system which defines the universal group $U(P)$. Conversely, if S is a triangular geodesic group system then P_S is a pregroup, whose universal group is that defined by S.

Characterization of pregroups in terms of geodesic systems

We say that a rewriting system $S \subseteq \Gamma^* \times \Gamma^*$ is *triangular* if each rule $\ell \to r \in S$ satisfies the "triangular" condition: $|\ell| = 2$, $|r| \leq 1$, so every rule in S is of the form $ab \to c$ where $a, b \in \Gamma$ and $c \in \Gamma \cup \{1\}$.

Theorem

Let P be a pregroup. Then the reduced part of the rewriting system $S'(P)$ is a geodesic triangular group system which defines the universal group $U(P)$. Conversely, if S is a triangular geodesic group system then P_S is a pregroup, whose universal group is that defined by S.