Computational problems in infinite groups

Laura Ciobanu — Université de Neuchâtel

Volker Diekert — Universität Stuttgart

Murray Elder — The University of Newcastle Australia

Andrew Rechnitzer — UBC

Buks van Rensburg — York Université University

Mathematics and Computation, CARMA June 2015

Part I: Equations in free groups

Let $A = \{a_1, a_1^{-1}, \ldots, a_d, a_d^{-1}\}$.

Let $R_A \subset A^*$ be the set of all words that do not contain any $a_i a_i^{-1}$ or $a_i^{-1} a_i$ pair. We call such words *reduced*.

The *free group* on A, denoted F_A, is the set of all reduced words with the operation of *concatenate then reduce*.

Part I: Equations in free groups

Let $A = \{a_1, a_1^{-1}, \ldots, a_d, a_d^{-1}\}$.

Let $R_A \subset A^*$ be the set of all words that do not contain any $a_i a_i^{-1}$ or $a_i^{-1} a_i$ pair. We call such words *reduced*.

The *free group* on A, denoted F_A, is the set of all reduced words with the operation of *concatenate then reduce*.

Let $\Omega = \{X_1, X_1^{-1}, \ldots, X_s, X_s^{-1}\}$ be another set.

An *equation in F_A* is an expression $U = V$ where $U, V \in (A \cup \Omega)^*$.

Part I: Equations in free groups

Let $A = \{a_1, a_1^{-1}, \ldots, a_d, a_d^{-1}\}$.

Let $R_A \subset A^*$ be the set of all words that do not contain any $a_i a_i^{-1}$ or $a_i^{-1} a_i$ pair. We call such words *reduced*.

The *free group* on A, denoted F_A, is the set of all reduced words with the operation of *concatenate then reduce*.

Let $\Omega = \{X_1, X_1^{-1}, \ldots, X_s, X_s^{-1}\}$ be another set.

An *equation in F_A* is an expression $U = V$ where $U, V \in (A \cup \Omega)^*$.

Eg: $a^{-1}X = Yb$ $\qquad\qquad$ $aXXb = YYbX$

Part I: Equations in free groups

Let $A = \{a_1, a_1^{-1}, \dots, a_d, a_d^{-1}\}$.

Let $R_A \subset A^*$ be the set of all words that do not contain any $a_i a_i^{-1}$ or $a_i^{-1} a_i$ pair. We call such words *reduced*.

The *free group* on A, denoted F_A, is the set of all reduced words with the operation of *concatenate then reduce*.

Let $\Omega = \{X_1, X_1^{-1}, \dots, X_s, X_s^{-1}\}$ be another set.

An *equation in F_A* is an expression $U = V$ where $U, V \in (A \cup \Omega)^*$.

Eg: $a^{-1}X = Yb$ $\qquad$ $aXXb = YYbX$

A *solution* to the equation is a map $X_j \mapsto u_j, X_j^{-1} \mapsto u_j^{-1}$ with $u_j \in R_A$ which makes $U = V$ true in F_A.

Equations in free groups

Eg: $aXXb = YYbX$

$$Y \to aY \qquad aXXb = aYaYbX$$

$$XXb = YaYbX$$

$$X \to Xb^{-1} \qquad Xb^{-1}Xb^{-1}b = YaYbXb^{-1}$$

$$Xb^{-1}X = YaYbXb^{-1}$$

$$\vdots$$

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

1987 Razborov: algorithmic description of all solutions (using Makanin)

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

1987 Razborov: algorithmic description of all solutions (using Makanin)

1990 Koscielski and Pacholski: Makanin's scheme not primitive recursive

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

1987 Razborov: algorithmic description of all solutions (using Makanin)

1990 Koscielski and Pacholski: Makanin's scheme not primitive recursive

1998 Plandowski: new approach to solving equations over *free monoids* in PSPACE using *data compression*

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

1987 Razborov: algorithmic description of all solutions (using Makanin)

1990 Koscielski and Pacholski: Makanin's scheme not primitive recursive

1998 Plandowski: new approach to solving equations over *free monoids* in PSPACE using *data compression*

2000 Gutierrez: solving equations over free groups in PSPACE

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

1987 Razborov: algorithmic description of all solutions (using Makanin)

1990 Koscielski and Pacholski: Makanin's scheme not primitive recursive

1998 Plandowski: new approach to solving equations over *free monoids* in PSPACE using *data compression*

2000 Gutierrez: solving equations over free groups in PSPACE

2013 Jez: new *recompression* technique – simplied all previous proofs

History

1982 Makanin: algorithm to decide if an equation in a free group has a solution

Algorithm extremely inefficient, time complexity DTIME $\left(2^{2^{2^{2^{poly(n)}}}}\right)$

1987 Razborov: algorithmic description of all solutions (using Makanin)

1990 Koscielski and Pacholski: Makanin's scheme not primitive recursive

1998 Plandowski: new approach to solving equations over *free monoids* in PSPACE using *data compression*

2000 Gutierrez: solving equations over free groups in PSPACE

2013 Jez: new *recompression* technique – simplied all previous proofs

2014 Diekert, Jez and Plandowski: NSPACE(n^2) algorithm

How complicated are solution sets?

We were interested in understanding the set of solutions to an equation as a formal language

$$\{u_1\#\ldots\#u_s \mid u_j \in R_A, X_j \mapsto u_j \text{ solves } U = V\}.$$

How complicated are solution sets?

We were interested in understanding the set of solutions to an equation as a formal language

$$\{u_1\# \ldots \# u_s \mid u_j \in R_A, X_j \mapsto u_j \text{ solves } U = V\}.$$

We asked how complicated solutions sets can be, in terms of formal languages.

How complicated are solution sets?

We were interested in understanding the set of solutions to an equation as a formal language

$$\{u_1\# \dots \# u_s \mid u_j \in R_A, X_j \mapsto u_j \text{ solves } U = V\}.$$

We asked how complicated solutions sets can be, in terms of formal languages.

It is clear that the set of solutions to any equation is recognised by a linear bounded automaton so solutions are context-sensitive.

But they are even simpler:

How complicated are solution sets?

We were interested in understanding the set of solutions to an equation as a formal language

$$\{u_1\# \dots \#u_s \mid u_j \in R_A, X_j \mapsto u_j \text{ solves } U = V\}.$$

We asked how complicated solutions sets can be, in terms of formal languages.

It is clear that the set of solutions to any equation is recognised by a linear bounded automaton so solutions are context-sensitive.

But they are even simpler:

Theorem (Ciobanu, Diekert, E 2015)

The set of solutions in reduced words to an equation in a free group is EDT0L.

How complicated are solution sets?

Theorem (Ciobanu, Diekert, E 2015)

The set of solutions in reduced words to an equation in a free group is EDT0L.

More precisely, we prove that the set of solutions is equal to

$$\{h(\#) \mid h \in R\}$$

where R is a regular language of endomorphisms over $C \supseteq A \cup \{\#\}$,

the NFA accepting R can be constructed in $\mathrm{NSPACE}(n \log n)$,

and the equation has zero/finitely many solutions iff the NFA has no accept state/no cycle.

How complicated are solution sets?

Theorem (Ciobanu, Diekert, E 2015)

The set of solutions in reduced words to an equation in a free group is EDT0L.

This is a surprising result – before Makanin it was thought the problem could be undecidable,

and trying to obtain actual solutions by following Makanin-like schemes (eg. using *Makanin-Razborov diagrams*) seems pretty hopeless.

Our result says you can describe the set of all solutions in a particularly easy way.

Outline of the proof

1. Reduce to the problem of finding all solutions to an equation over a *free monoid with involution* over $A = \{a_i, \overline{a_i}\}$.

Outline of the proof

1. Reduce to the problem of finding all solutions to an equation over a *free monoid with involution* over $A = \{a_{\mathrm{i}}, \overline{a_{\mathrm{i}}}\}$.

2. Construct a finite graph with vertices labeled by *extended equations*, and edges which enable the following moves between them:

   – pop variables $X \to aX, \overline{X} \to \overline{X}\overline{a}$ or $X, \overline{X} \to 1$ (equation length grows)

   – compress pairs of constants $ab \to c$ (equation length shrinks,

   – compress blocks of constants $aa \dots a \to a_\ell$ alphabet of constants grows)

Outline of the proof

1. Reduce to the problem of finding all solutions to an equation over a *free monoid with involution* over $A = \{a_i, \overline{a_i}\}$.

2. Construct a finite graph with vertices labeled by *extended equations*, and edges which enable the following moves between them:

   – pop variables $X \to aX, \overline{X} \to \overline{X}\overline{a}$ or $X, \overline{X} \to 1$ (equation length grows)

   – compress pairs of constants $ab \to c$ (equation length shrinks,

   – compress blocks of constants $aa \ldots a \to a_\ell$ alphabet of constants grows)

   We fix an enlarged set of constants with involution $C \supset A$ of size $O(|UV| + |A|)$ and restrict to extended equations over $C \cup \Omega$ of length at most a fixed bound in $O(|UV| + |A|)$.

   This guarantees the graph is finite. We must prove that the graph encodes all solutions with these restrictions.

Proving the graph contains solutions

We define an *initial vertex* to be labeled by an extended equation with $U = V$, and a *final vertex* to be labeled by an extended equation with $P = P$ where $P \in C^*$ (ie. no variables).

Proving the graph contains solutions

We define an *initial vertex* to be labeled by an extended equation with $U = V$, and a *final vertex* to be labeled by an extended equation with $P = P$ where $P \in C^*$ (ie. no variables).

Edges are defined so that a solution to the target extended equation implies a solution to the source extended equation.

Proving the graph contains solutions

We define an *initial vertex* to be labeled by an extended equation with $U = V$, and a *final vertex* to be labeled by an extended equation with $P = P$ where $P \in C^*$ (ie. no variables).

Edges are defined so that a solution to the target extended equation implies a solution to the source extended equation.

It follows that a path from initial to final vertices encodes a sequence of moves on the equation $U = V$ converting it to an equation $P = P$ with no variables. Since $P = P$ has a solution, it can be carried back to a solution for $U = V$.

Proving the graph contains *all* solutions

Suppose we *know* a solution $X_j \to u_i$ for $U = V$.

Since we know what the solution is, we could simply follow $X \to aX$, $X \to 1$ edges until all variables disappear, and arrive at a final vertex.

Proving the graph contains *all* solutions

Suppose we *know* a solution $X_j \to u_i$ for $U = V$.

Since we know what the solution is, we could simply follow $X \to aX$, $X \to 1$ edges until all variables disappear, and arrive at a final vertex.

But this would take us out of the graph.

Proving the graph contains *all* solutions

Suppose we *know* a solution $X_j \to u_i$ for $U = V$.

Since we know what the solution is, we could simply follow $X \to aX$, $X \to 1$ edges until all variables disappear, and arrive at a final vertex.

But this would take us out of the graph.

Note that as we pop, the number of variables in the equation never increases, and the substrings of constants in between them grow in length.

Applying the compression moves shrinks the equation back down, so we can continue popping to find a solution.

Proving the graph contains *all* solutions

Here is the procedure. Put $n = |UV| + |A|$.

- Assume equation length is at most $29n$ (true for an initial vertex).

Proving the graph contains *all* solutions

Here is the procedure. Put $n = |UV| + |A|$.

- Assume equation length is at most $29n$ (true for an initial vertex).
- Pop the first and last letter from each variable. Now the equation length is at most $31n$.

Proving the graph contains *all* solutions

Here is the procedure. Put $n = |UV| + |A|$.

- Assume equation length is at most $29n$ (true for an initial vertex).
- Pop the first and last letter from each variable. Now the equation length is at most $31n$.
- Run a subroutine (*block compression*) to replace maximal blocks $aa \dots a$ by a. Now the equation has no aa factors, and length at most $31n$.

Proving the graph contains *all* solutions

Here is the procedure. Put $n = |UV| + |A|$.

- Assume equation length is at most $29n$ (true for an initial vertex).
- Pop the first and last letter from each variable. Now the equation length is at most $31n$.
- Run a subroutine (*block compression*) to replace maximal blocks $aa \ldots a$ by a. Now the equation has no aa factors, and length at most $31n$.
- Run a subroutine (*pair compression*) which applies the moves $ab \to c$ in a careful way to ensure that the equation length reduces to at most $29n$. Repeat.

Proving the graph contains *all* solutions

Here is the procedure. Put $n = |UV| + |A|$.

- Assume equation length is at most $29n$ (true for an initial vertex).
- Pop the first and last letter from each variable. Now the equation length is at most $31n$.
- Run a subroutine (*block compression*) to replace maximal blocks $aa \dots a$ by a. Now the equation has no aa factors, and length at most $31n$.
- Run a subroutine (*pair compression*) which applies the moves $ab \to c$ in a careful way to ensure that the equation length reduces to at most $29n$. Repeat.

Each round reduces the *weight* of the solution (defined as the sum of the length of words substituted for each variable) so the procedure terminates at a final vertex.

More details in our paper to appear in ICALP2015 proceedings, longer version to appear IJAC.

Part II: Cogrowth

Let G be a group with presentation $\langle S \mid R \rangle$, so $G \cong F_S / \langle\langle R \rangle\rangle$.

Eg: $\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$ $\qquad F_{\{a,b,a^{-1},b^{-1}\}} = \langle a, b \mid - \rangle$

Part II: Cogrowth

Let G be a group with presentation $\langle S \mid R \rangle$, so $G \cong F_S/\langle\langle R \rangle\rangle$.

Eg: $\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$ $\qquad$ $F_{\{a,b,a^{-1},b^{-1}\}} = \langle a, b \mid - \rangle$

Define $c_n = \#$ words in $\langle\langle R \rangle\rangle$ of length n. The function $n \mapsto c_n$ is called the *cogrowth function* for (G, S).

$c_n \leq (2|S|)(2|S| - 1)^{n-1}$ $\qquad$ so $\qquad$ $\limsup c_n^{1/n} \leq 2|S| - 1$

Part II: Cogrowth

Let G be a group with presentation $\langle S \mid R \rangle$, so $G \cong F_S / \langle\!\langle R \rangle\!\rangle$.

Eg: $\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$ $\qquad F_{\{a,b,a^{-1},b^{-1}\}} = \langle a, b \mid - \rangle$

Define $c_n = \#$ words in $\langle\!\langle R \rangle\!\rangle$ of length n. The function $n \mapsto c_n$ is called the *cogrowth function* for (G, S).

$c_n \leq (2|S|)(2|S| - 1)^{n-1}$ $\qquad$ so $\qquad$ $\limsup c_n^{1/n} \leq 2|S| - 1$

Theorem (Grigorchuk/Cohen)

Let G be a non-free group. G is amenable *iff* $\limsup c_n^{1/n} = 2|S| - 1$.

Cogrowth

The formal power series $f(z) = \sum c_n z^n$ is the *cogrowth series*.

$\limsup c_n^{1/n}$ is the reciprocal of the radius of convergence of $f(z)$. We call this number the *cogrowth*.

Example

$G = \langle a \mid a^2 \rangle$. We consider a, a^{-1} as distinct formal symbols, so:

Example

$G = \langle a \mid a^2 \rangle$. We consider a, a^{-1} as distinct formal symbols, so:

$c_{2n+1} = 0$ since the relator has even length

$$\begin{array}{lcll} c_0 & = & 1 & \epsilon \\ c_2 & = & 2 & aa,\ a^{-1}a^{-1} \\ c_4 & = & 2 & aaaa,\ a^{-1}a^{-1}a^{-1}a^{-1} \\ & \vdots & & \end{array}$$

Example

$G = \langle a \mid a^2 \rangle$. We consider a, a^{-1} as distinct formal symbols, so:

$c_{2n+1} = 0$ since the relator has even length

$$\begin{array}{rclcl} c_0 & = & 1 & & \epsilon \\ c_2 & = & 2 & & aa,\ a^{-1}a^{-1} \\ c_4 & = & 2 & & aaaa,\ a^{-1}a^{-1}a^{-1}a^{-1} \\ & \vdots & & & \end{array}$$

$$f(z) \;=\; 1 + \sum_{n=1}^{\infty} 2z^{2n} \;=\; 1 + 2\left(\frac{1}{1-z^2} - 1\right) \;=\; \frac{1+z^2}{1-z^2}$$

(radius of convergence is 1)

Example

$G = \langle a \mid a^2 \rangle$. We consider a, a^{-1} as distinct formal symbols, so:

$c_{2n+1} = 0$ since the relator has even length

$$\begin{array}{rcl l} c_0 & = & 1 & \epsilon \\ c_2 & = & 2 & aa,\ a^{-1}a^{-1} \\ c_4 & = & 2 & aaaa,\ a^{-1}a^{-1}a^{-1}a^{-1} \\ & \vdots & & \end{array}$$

$$f(z) = 1 + \sum_{n=1}^{\infty} 2z^{2n} = 1 + 2\left(\frac{1}{1-z^2} - 1\right) = \frac{1+z^2}{1-z^2}$$

(radius of convergence is 1)

Theorem (Kouksov)

Let G be a non-free group. The cogrowth series is rational iff G is finite.

Alternative

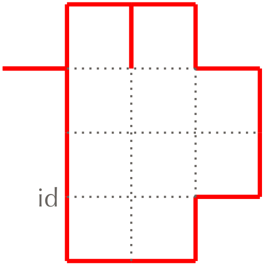
$d_n = \#$ **all** words in $(S \cup S^{-1})^*$ equal to id

There are formulas to switch between d_n and c_n at the level of generating functions

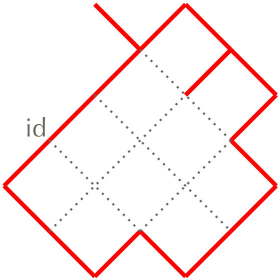
Theorem (Grigorchuk/Cohen)

G is amenable iff $\limsup d_n^{1/n} = 2|S|$.

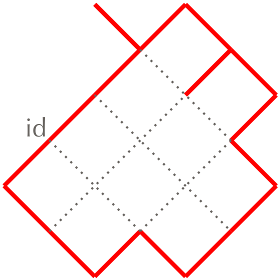
$$\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$$



$$\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$$

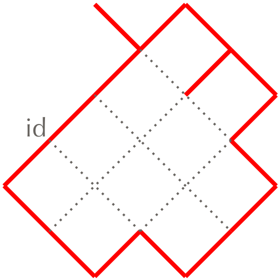


$\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$



a	a	b^{-1}	b	a	b	a^{-1}	a	b	a^{-1}	b	a^{-1}	a^{-1}	b^{-1}	a^{-1}	b^{-1}	b^{-1}	a
$+$	$+$	$-$	$+$	$+$	$+$	$-$	$+$	$+$	$-$	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$+$	$+$	$+$	$-$	$+$	$-$	$-$	$+$	$-$	$-$	$-$	$-$	$-$	$+$	$-$	$+$	$+$	$+$

$\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$



a	a	b^{-1}	b	a	b	a^{-1}	a	b	a^{-1}	b	a^{-1}	a^{-1}	b^{-1}	a^{-1}	b^{-1}	b^{-1}	a
$+$	$+$	$-$	$+$	$+$	$+$	$-$	$+$	$+$	$-$	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$+$	$+$	$+$	$-$	$+$	$-$	$-$	$+$	$-$	$-$	$-$	$-$	$-$	$+$	$-$	$+$	$+$	$+$

$d_{2n} = \binom{2n}{n}\binom{2n}{n}$ $\qquad$ $d_{2n+1} = 0$

Open problem for amenability

A famous open problem is whether one particular (infamous) example is amenable:

Richard Thompson's group $F = \langle a, b \mid [ab^{-1}, a^{-1}ba], [ab^{-1}, a^{-2}ba^2] \rangle$

Open problem for amenability

A famous open problem is whether one particular (infamous) example is amenable:

Richard Thompson's group $F = \langle a, b \mid [ab^{-1}, a^{-1}ba], [ab^{-1}, a^{-2}ba^2] \rangle$

Several authors have tried computational approaches to the problem:

- Burillo, Cleary, Weist 2007
- Arzhantseva, Guba, Lustig, Préaux 2008
- E, Rechnitzer, Wong 2012
- Haagarup, Haagarup, Ramirez-Solano 2015

New method: "Random walk on the set of trivial words"

$G = \langle S \mid R \rangle$

$\mathcal{X} = \langle\langle R \rangle\rangle$

$\mathcal{R} = \{$all freely reduced cyclic permutations of words in $R \cup R^{-1}\}$

$P : \mathcal{R} \to [0, 1]$ a prob dist st $P(r) > 0$ and $P(r) = P(r^{-1})$ for all $r \in \mathcal{R}$

New method: "Random walk on the set of trivial words"

$G = \langle S \mid R \rangle$

$\mathcal{X} = \langle\langle R \rangle\rangle$

$\mathcal{R} = \{$all freely reduced cyclic permutations of words in $R \cup R^{-1}\}$

$P : \mathcal{R} \to [0, 1]$ a prob dist st $P(r) > 0$ and $P(r) = P(r^{-1})$ for all $r \in \mathcal{R}$

We define a *Markov chain* with states $\mathcal{X}$ and transitions between states (trivial words) with prescribed probabilities

New method: “Random walk on the set of trivial words”

$G = \langle S \mid R \rangle$

$\mathcal{X} = \langle\langle R \rangle\rangle$

$\mathcal{R} = \{$all freely reduced cyclic permutations of words in $R \cup R^{-1}\}$

$P : \mathcal{R} \to [0, 1]$ a prob dist st $P(r) > 0$ and $P(r) = P(r^{-1})$ for all $r \in \mathcal{R}$

We define a *Markov chain* with states $\mathcal{X}$ and transitions between states (trivial words) with prescribed probabilities

so that moves are “uniquely reversible” and there is a positive probability of reaching every state from any given state.

Moves on $w \in \mathcal{X}$

1. *Conjugation* by $x \in S \cup S^{-1}$:
   - write xwx^{-1} and freely reduce to obtain w'.

2. *Left-insertion* by $r \in \mathcal{R}$ at position $m \in [0, |w|]$:
   - write $w = uv$ with $|v| = m$
   - freely reduce ur to obtain u'
   - freely reduce $u'v$ to obtain w'
   - if any part of v is cancelled, set $w' = w$ (reject the move)

Lemma

Each move is uniquely reversible in the following sense:

conjugation by $x \longleftrightarrow$ conjugation by x^{-1}

left-insertion of r at $m \longleftrightarrow$ left-insertion of r^{-1} at m

Metropolis MCMC algorithm

Next, we define the transition probabilities for our Markov chain. Let $p_c, \beta \in (0, 1)$ and $\alpha \in \mathbb{R}$ be parameters.

Let w_n be the current word. Construct the next word w_{n+1} as follows:

1. With probability p_c, choose to do a conjugation, else $(1 - p_c)$ choose an insertion.

2. If conjugation, choose $s \in S \cup S^{-1}$ with probability $\frac{1}{2|S|}$, and conjugate to obtain w'.

   With probability $\min\left\{1, \frac{(|w'|+1)^{1+\alpha}}{(|w|+1)^{1+\alpha}} \cdot \frac{\beta^{|w'|}}{\beta^{|w|}}\right\}$ put $w_{n+1} = w'$.

   Else put $w_{n+1} = w_n$ (reject the move)

Metropolis MCMC algorithm

3. If left-insertion, choose $r \in \mathcal{R}$ with probability $P(r)$ and $m \in [0, |w|]$ with uniform probability. Left-insert to obtain w'.

   With probability $\min\left\{1, \frac{(|w'|+1)^\alpha}{(|w|+1)^\alpha} \cdot \frac{\beta^{|w'|}}{\beta^{|w|}}\right\}$ put $w_{n+1} = w'$.

   Else put $w_{n+1} = w_n$ (reject the move)

Metropolis MCMC algorithm

3. If left-insertion, choose $r \in \mathcal{R}$ with probability $P(r)$ and $m \in [0, |w|]$ with uniform probability. Left-insert to obtain w'.

   With probability $\min\left\{1, \frac{(|w'|+1)^\alpha}{(|w|+1)^\alpha} \cdot \frac{\beta^{|w'|}}{\beta^{|w|}}\right\}$ put $w_{n+1} = w'$.

   Else put $w_{n+1} = w_n$ (reject the move)

Eg: $G = \langle a \mid a^2 \rangle$. The state space $\mathcal{X}$ is

$$\rightsquigarrow a^{-4} \rightsquigarrow a^{-2} \rightsquigarrow \epsilon \rightsquigarrow a^{2} \rightsquigarrow a^{4} \rightsquigarrow$$

Starting at $w_0 = a^{2k}$

conjugation: no change

left-insert $a^{\pm 2}$: move left/right (or no change if rejected)

Metropolis MCMC algorithm

The probabilities are chosen specifically so that we can prove that:

> there is a probability distribution π on $\mathcal{X}$ such that the probability that the algorithm reaches state w after N steps converges to $\pi(w)$.

Metropolis MCMC algorithm

The probabilities are chosen specifically so that we can prove that:

> there is a probability distribution π on $\mathcal{X}$ such that the probability that the algorithm reaches state w after N steps converges to $\pi(w)$.

If $\Pr(u \to v)$ is the probability of moving from u to v in one step, a distribution is *stationary* for the MC if $\pi(u) = \sum_v \Pr(v \to u)\pi(v)$.

Metropolis MCMC algorithm

The probabilities are chosen specifically so that we can prove that:

> there is a probability distribution π on $\mathcal{X}$ such that the probability that the algorithm reaches state w after N steps converges to $\pi(w)$.

If $\Pr(u \to v)$ is the probability of moving from u to v in one step, a distribution is *stationary* for the MC if $\pi(u) = \sum_v \Pr(v \to u)\pi(v)$.

Theorem (E, Rechnitzer, van Rensburg)

$$\pi(w) = \frac{(|w| + 1)^{1+\alpha}\beta^{|w|}}{Z}$$

where Z is a normalising constant is the unique stationary distribution for the algorithm.

i.e. the probability that the algorithm reaches state w after N steps converges to $\pi(w)$.

Relation to cogrowth

Since π is a probability distribution we have $\sum_{w \in \mathcal{X}} \pi(w) = 1$

so since $\pi(w) = \frac{(|w|+1)^{1+\alpha} \beta^{|w|}}{Z}$ we get

$$Z = \sum_{w \in \mathcal{X}} (|w|+1)^{1+\alpha} \beta^{|w|} \qquad = \sum_n c_n (n+1)^{1+\alpha} \beta^n$$

which converges when β is less than the radius of convergence of the cogrowth series ($=$ recip of cogrowth rate)

Relation to cogrowth

The mean length of a word sampled by running the algorithm is

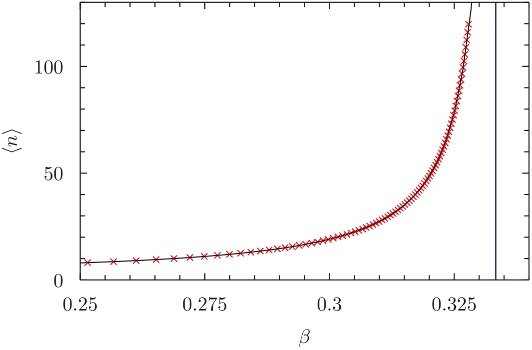
$$E(|w|) = \sum_{u \in \mathcal{X}} |u| \pi(u) \qquad = \sum_{u \in \mathcal{X}} |u| \frac{(|u|+1)^{1+\alpha} \beta^{|u|}}{Z}$$

$$= \sum_{n} \frac{n(n+1)^{1+\alpha} \beta^{n} c_n}{Z}$$

$$= \frac{\sum_n n(n+1)^{1+\alpha} \beta^{n} c_n}{\sum_n (n+1)^{1+\alpha} \beta^{n} c_n}$$

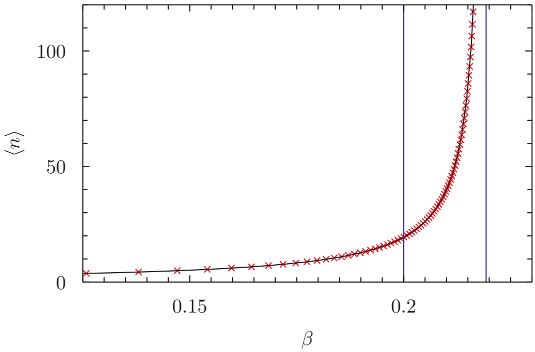
As $\beta \to$ recip of cogrowth rate (from below), the mean length $\to +\infty$

$\mathbb{Z}^2 = \langle a, b \mid aba^{-1}b^{-1} \rangle$



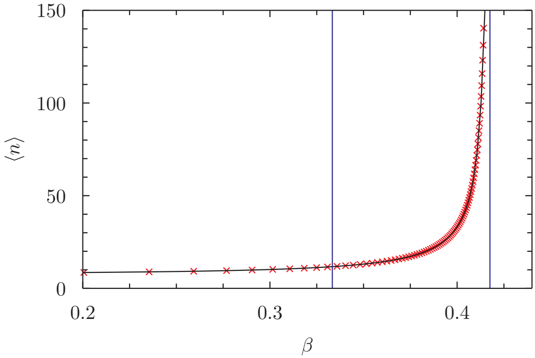
The mean length of sampled words plotted against β for $\langle a, b \mid aba^{-1}b^{-1} \rangle$ with $\alpha = 1$. The crosses indicate data obtained from an implementation of the algorithm while the curve indicates the expectation derived from the exact cogrowth series for the group. The vertical line indicates $\beta_c = 1/3$.

Free product $\langle a, b, c | a^2, b^2, c^2 \rangle$



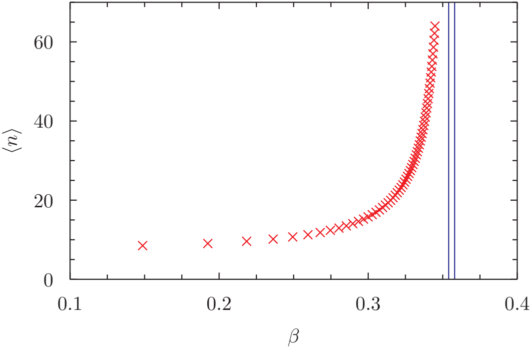
Mean length of sampled words vs. β for $\langle a, b, c | a^2, b^2, c^2 \rangle$ sampled with $\alpha = 1$. The crosses indicate data obtained from the algorithm, while the curves indicates the expectation derived from the exact cogrowth series (found by Kouksov). Vertical lines at $1/5$ and 0.2192752634 (the reciprocal of the cogrowth).

Baumslag-Solitar(3,3)



Mean length of sampled words vs. β for $BS(3,3) = \langle a, b \mid ba^3b^{-1}a^{-3} \rangle$ with $\alpha = 1$. The crosses indicate data obtained from the algorithm, while the curves indicates the expectation derived from the (known) cogrowth series for the group (found by E, Rechnitzer, van Rensburg and Wong). Vertical lines at $1/3$ and 0.417525628 (the reciprocal of the cogrowth).

Surface group $\langle a, b, c, d \mid [a, b][c, d] \rangle$



Mean length sampled words vs. β for the presentation $\langle a, b, c, d \mid [a, b][c, d] \rangle$ for $\alpha = 1$. Blue lines indicate bounds upper and lower bounds 0.35473, 0.3547 proven by Nagnibeda and Gouëzel.

Thompson's group F

Let $F = \langle a, b \mid [ab^{-1}, a^{-1}ba], [ab^{-1}, a^{-2}ba^2] \rangle$

$|S| = 2$ so if F is amenable, $\limsup c_n^{1/n} = 3$

and we would expect the mean length of words sampled by the MC algorithm to blow up at $1/3$.

Let's run the algorithm . . .

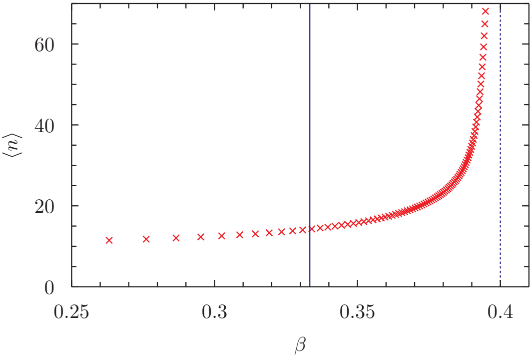
Thompson's group F

Let $F = \langle a, b \mid [ab^{-1}, a^{-1}ba], [ab^{-1}, a^{-2}ba^2] \rangle$

$|S| = 2$ so if F is amenable, $\limsup c_n^{1/n} = 3$

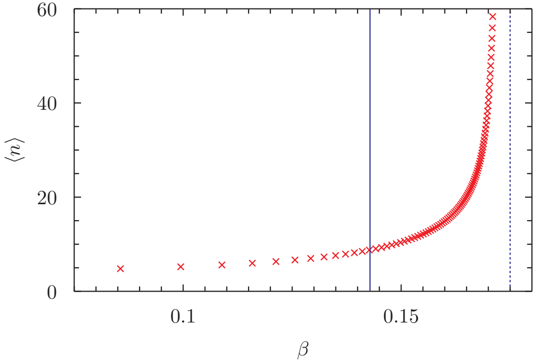
and we would expect the mean length of words sampled by the MC algorithm to blow up at $1/3$.

Let's run the algorithm ...



Thompson's group F

Another presentation
$\langle a, b, c, d \mid c = a^{-1}ba, d = a^{-1}ca, [ab^{-1}, c], [ab^{-1}, d] \rangle$ for F



Will this method ever give a proof?

There are very few examples of Markov chain algorithms for which one can *prove* rate of convergence to the stationary distribution.

Will this method ever give a proof?

There are very few examples of Markov chain algorithms for which one can *prove* rate of convergence to the stationary distribution.

Eg: shuffle a deck of cards by selecting a card at random and placing it at the bottom of the deck

Start with  on the bottom of the deck, when  is on top, stop – uniform distribution.

Will this method ever give a proof?

There are very few examples of Markov chain algorithms for which one can *prove* rate of convergence to the stationary distribution.

Eg: shuffle a deck of cards by selecting a card at random and placing it at the bottom of the deck

Start with  on the bottom of the deck, when  is on top, stop – uniform distribution.

Cameron Rogers is exploring pathalogical cases that break the algorithm

Eg: $\langle a, b \mid abab^{-1}a^{-1}b^{-1}, a^n b \rangle$

as well as connections between convergence of the walk and complexity of the Følner function.

More details in our paper to appear in Experimental Mathematics.