The central tree property and some average case complexity results for algorithmic problems in free groups

Enric Ventura

Departament de Matemàtiques
Universitat Politècnica de Catalunya

New York Group Theory Seminar

Complexity Day

(joint work with M. Roy and P. Weil)

December 8th, 2023.

Outline

Outline

1. Complexity of algorithms
2. On Whitehead's algorithm
3. The Central Tree Property

Decision problems

Definition

A decision problem $\mathcal{P}$ is determined by a well-defined set of inputs I, and a $\texttt{YES/NO}$ *property $P \subseteq I$ you want to know about each of them:*

- *Given $u \in I$,*
- *Decide whether u satisfies P (i.e., $u \in P$).*

Typically, the set I comes with a notion of size (or length), $\ell \colon I \to \mathbb{N}$, such that, for every $n \geqslant 0$, $|\{u \in I \mid \ell(u) \leqslant n\}| < \infty$.

Definition

A decision problem is solvable when there exists an algorithm $\mathcal{A}$ (i.e., a Turing machine) answering correctly for each given input $u \in I$.

Decision problems

Definition

A decision problem $\mathcal{P}$ is determined by a well-defined set of inputs I, and a YES/NO *property $P \subseteq I$ you want to know about each of them:*

- *Given $u \in I$,*
- *Decide whether u satisfies P (i.e., $u \in P$).*

Typically, the set I comes with a notion of size (or length), $\ell \colon I \to \mathbb{N}$, such that, for every $n \geqslant 0$, $|\{u \in I \mid \ell(u) \leqslant n\}| < \infty$.

Definition

A decision problem is solvable when there exists an algorithm $\mathcal{A}$ (i.e., a Turing machine) answering correctly for each given input $u \in I$.

Decision problems

Definition

A decision problem $\mathcal{P}$ is determined by a well-defined set of inputs I, and a $\mathtt{YES/NO}$ property $P \subseteq I$ you want to know about each of them:

- *Given $u \in I$,*
- *Decide whether u satisfies P (i.e., $u \in P$).*

Typically, the set I comes with a notion of size (or length), $\ell \colon I \to \mathbb{N}$, such that, for every $n \geqslant 0$, $|\{u \in I \mid \ell(u) \leqslant n\}| < \infty$.

Definition

A decision problem is solvable when there exists an algorithm $\mathcal{A}$ (i.e., a Turing machine) answering correctly for each given input $u \in I$.

Complexity of algorithms

Definition

Suppose algorithm $\mathcal{A}$ solves a decision problem $\mathcal{P}$.

- *Given $u \in I$, we denote by $\mathtt{t}(u)$ the time (i.e., number of steps) taken by $\mathcal{A}$ to give the correct answer for input u.*
- *The worst case complexity of $\mathcal{A}$ is the function $\mathtt{wc}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{wc}_{\mathcal{A}}(n) = \max_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)$.*
- *The average case complexity of $\mathcal{A}$ is the function $\mathtt{ac}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{ac}_{\mathcal{A}}(n) = \frac{\sum_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)}{|\{u \in I \mid \ell(u) \leqslant n\}|}$.*
- *These functions are only interesting up to asymptotic equivalence.*

Observation

Clearly, $\mathtt{ac}_{\mathcal{A}}(n) \leqslant \mathtt{wc}_{\mathcal{A}}(n)$. But ... there are cases where $\mathtt{ac}_{\mathcal{A}}(n)$ is much smaller than $\mathtt{wc}_{\mathcal{A}}(n)$ ...

Complexity of algorithms

Definition

Suppose algorithm $\mathcal{A}$ solves a decision problem $\mathcal{P}$.

- *Given $u \in I$, we denote by $\mathtt{t}(u)$ the time (i.e., number of steps) taken by $\mathcal{A}$ to give the correct answer for input u.*
- *The worst case complexity of $\mathcal{A}$ is the function $\mathtt{wc}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{wc}_{\mathcal{A}}(n) = \max_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)$.*
- *The average case complexity of $\mathcal{A}$ is the function $\mathtt{ac}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{ac}_{\mathcal{A}}(n) = \frac{\sum_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)}{|\{u \in I \mid \ell(u) \leqslant n\}|}$.*
- *These functions are only interesting up to asymptotic equivalence.*

Observation

Clearly, $\mathtt{ac}_{\mathcal{A}}(n) \leqslant \mathtt{wc}_{\mathcal{A}}(n)$. But ... there are cases where $\mathtt{ac}_{\mathcal{A}}(n)$ is much smaller than $\mathtt{wc}_{\mathcal{A}}(n)$ ...

Complexity of algorithms

Definition

Suppose algorithm $\mathcal{A}$ solves a decision problem $\mathcal{P}$.

- *Given $u \in I$, we denote by $\mathtt{t}(u)$ the time (i.e., number of steps) taken by $\mathcal{A}$ to give the correct answer for input u.*
- *The worst case complexity of $\mathcal{A}$ is the function $\mathtt{wc}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{wc}_{\mathcal{A}}(n) = \max_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)$.*
- *The average case complexity of $\mathcal{A}$ is the function $\mathtt{ac}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{ac}_{\mathcal{A}}(n) = \frac{\sum_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)}{|\{u \in I \mid \ell(u) \leqslant n\}|}$.*
- *These functions are only interesting up to asymptotic equivalence.*

Observation

Clearly, $\mathtt{ac}_{\mathcal{A}}(n) \leqslant \mathtt{wc}_{\mathcal{A}}(n)$. But ... there are cases where $\mathtt{ac}_{\mathcal{A}}(n)$ is much smaller than $\mathtt{wc}_{\mathcal{A}}(n)$ ...

Complexity of algorithms

Definition

Suppose algorithm $\mathcal{A}$ solves a decision problem $\mathcal{P}$.

- *Given $u \in I$, we denote by $\mathtt{t}(u)$ the time (i.e., number of steps) taken by $\mathcal{A}$ to give the correct answer for input u.*
- *The worst case complexity of $\mathcal{A}$ is the function $\mathtt{wc}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{wc}_{\mathcal{A}}(n) = \max_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)$.*
- *The average case complexity of $\mathcal{A}$ is the function $\mathtt{ac}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{ac}_{\mathcal{A}}(n) = \frac{\sum_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)}{|\{u \in I \mid \ell(u) \leqslant n\}|}$.*
- *These functions are only interesting up to asymptotic equivalence.*

Observation

Clearly, $\mathtt{ac}_{\mathcal{A}}(n) \leqslant \mathtt{wc}_{\mathcal{A}}(n)$. But ... there are cases where $\mathtt{ac}_{\mathcal{A}}(n)$ is much smaller than $\mathtt{wc}_{\mathcal{A}}(n)$ ...

Complexity of algorithms

Definition

Suppose algorithm $\mathcal{A}$ solves a decision problem $\mathcal{P}$.

- *Given $u \in I$, we denote by $\mathtt{t}(u)$ the time (i.e., number of steps) taken by $\mathcal{A}$ to give the correct answer for input u.*
- *The worst case complexity of $\mathcal{A}$ is the function $\mathtt{wc}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{wc}_{\mathcal{A}}(n) = \max_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)$.*
- *The average case complexity of $\mathcal{A}$ is the function $\mathtt{ac}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{ac}_{\mathcal{A}}(n) = \frac{\sum_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)}{|\{u \in I \mid \ell(u) \leqslant n\}|}$.*
- *These functions are only interesting up to asymptotic equivalence.*

Observation

Clearly, $\mathtt{ac}_{\mathcal{A}}(n) \leqslant \mathtt{wc}_{\mathcal{A}}(n)$. But ... there are cases where $\mathtt{ac}_{\mathcal{A}}(n)$ is much smaller than $\mathtt{wc}_{\mathcal{A}}(n)$ ...

Complexity of algorithms

Definition

Suppose algorithm $\mathcal{A}$ solves a decision problem $\mathcal{P}$.

• *Given $u \in I$, we denote by $\mathtt{t}(u)$ the time (i.e., number of steps) taken by $\mathcal{A}$ to give the correct answer for input u.*
• *The worst case complexity of $\mathcal{A}$ is the function $\mathtt{wc}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{wc}_{\mathcal{A}}(n) = \max_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)$.*
• *The average case complexity of $\mathcal{A}$ is the function $\mathtt{ac}_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$, $n \mapsto \mathtt{ac}_{\mathcal{A}}(n) = \frac{\sum_{\{u \in I \mid \ell(u) \leqslant n\}} \mathtt{t}(u)}{|\{u \in I \mid \ell(u) \leqslant n\}|}$.*
• *These functions are only interesting up to asymptotic equivalence.*

Observation

Clearly, $\mathtt{ac}_{\mathcal{A}}(n) \leqslant \mathtt{wc}_{\mathcal{A}}(n)$. But ... there are cases where $\mathtt{ac}_{\mathcal{A}}(n)$ is much smaller than $\mathtt{wc}_{\mathcal{A}}(n)$ ...

Average case complexity

A general idea to improve the average case complexity of $\mathcal{A}$:

- *Find a variant $\mathcal{A}'$ of $\mathcal{A}$ running 'fast' on a 'big' subset $E \subseteq I$;*
- *Consider the new algorithm $\mathcal{A}''$:*
  Given $u \in I$, if $u \in E$ run $\mathcal{A}'$ on u; otherwise run $\mathcal{A}$ on u.

(Except in degenerate cases,) we have $\mathrm{wc}_{\mathcal{A}''}(n) = \mathrm{wc}_{\mathcal{A}}(n)$ but it could very well be that $\mathrm{ac}_{\mathcal{A}''}(n) \ll \mathrm{ac}_{\mathcal{A}}(n)$.

This idea was recently exploited in the paper:

V. Shpilrain, Average-case complexity of the Whitehead problem for free groups. Comm. Algebra, 51(2) (2023), 799–806.

to get the following improvement of a classical result:

Average case complexity

A general idea to improve the average case complexity of $\mathcal{A}$:

- *Find a variant $\mathcal{A}'$ of $\mathcal{A}$ running 'fast' on a 'big' subset $E \subseteq I$;*
- *Consider the new algorithm $\mathcal{A}''$:*
  Given $u \in I$, if $u \in E$ run $\mathcal{A}'$ on u; otherwise run $\mathcal{A}$ on u.

(Except in degenerate cases,) we have $\mathrm{wc}_{\mathcal{A}''}(n) = \mathrm{wc}_{\mathcal{A}}(n)$ but it could very well be that $\mathrm{ac}_{\mathcal{A}''}(n) \ll \mathrm{ac}_{\mathcal{A}}(n)$.

This idea was recently exploited in the paper:

V. Shpilrain, Average-case complexity of the Whitehead problem for free groups. Comm. Algebra, 51(2) (2023), 799–806.

to get the following improvement of a classical result:

Average case complexity

A general idea to improve the average case complexity of $\mathcal{A}$:

- *Find a variant $\mathcal{A}'$ of $\mathcal{A}$ running 'fast' on a 'big' subset $E \subseteq I$;*
- *Consider the new algorithm $\mathcal{A}''$:*
  Given $u \in I$, if $u \in E$ run $\mathcal{A}'$ on u; otherwise run $\mathcal{A}$ on u.

(Except in degenerate cases,) we have $\mathrm{wc}_{\mathcal{A}''}(n) = \mathrm{wc}_{\mathcal{A}}(n)$ but it could very well be that $\mathrm{ac}_{\mathcal{A}''}(n) \ll \mathrm{ac}_{\mathcal{A}}(n)$.

This idea was recently exploited in the paper:

V. Shpilrain, Average-case complexity of the Whitehead problem for free groups. Comm. Algebra, 51(2) (2023), 799–806.

to get the following improvement of a classical result:

Average case complexity

A general idea to improve the average case complexity of $\mathcal{A}$:

- *Find a variant $\mathcal{A}'$ of $\mathcal{A}$ running 'fast' on a 'big' subset $E \subseteq I$;*
- *Consider the new algorithm $\mathcal{A}''$:*

  Given $u \in I$, if $u \in E$ run $\mathcal{A}'$ on u; otherwise run $\mathcal{A}$ on u.

(Except in degenerate cases,) we have $\mathtt{wc}_{\mathcal{A}''}(n) = \mathtt{wc}_{\mathcal{A}}(n)$ but it could very well be that $\mathtt{ac}_{\mathcal{A}''}(n) \ll \mathtt{ac}_{\mathcal{A}}(n)$.

This idea was recently exploited in the paper:

V. Shpilrain, Average-case complexity of the Whitehead problem for free groups. Comm. Algebra, 51(2) (2023), 799–806.

to get the following improvement of a classical result:

Average case complexity

A general idea to improve the average case complexity of $\mathcal{A}$:

- *Find a variant $\mathcal{A}'$ of $\mathcal{A}$ running 'fast' on a 'big' subset $E \subseteq I$;*
- *Consider the new algorithm $\mathcal{A}''$:*

  Given $u \in I$, if $u \in E$ run $\mathcal{A}'$ on u; otherwise run $\mathcal{A}$ on u.

(Except in degenerate cases,) we have $\mathtt{wc}_{\mathcal{A}''}(n) = \mathtt{wc}_{\mathcal{A}}(n)$ but it could very well be that $\mathtt{ac}_{\mathcal{A}''}(n) \ll \mathtt{ac}_{\mathcal{A}}(n)$.

This idea was recently exploited in the paper:

V. Shpilrain, Average-case complexity of the Whitehead problem for free groups. Comm. Algebra, 51(2) (2023), 799–806.

to get the following improvement of a classical result:

Outline

Classical Whitehead's algorithm

Theorem (Whitehead, 1936)

There is an algorithm $\mathcal{W}$ taking $w \in F_r$ as input, deciding whether w is primitive in F_r, and working in time $\mathrm{wc}_{\mathcal{W}}(n) = O(4^r r\, n^2) = O(n^2)$.

Observation

A given $w \in F_r$ is primitive $\Leftrightarrow \min_{\varphi \in \mathrm{Aut}(F_r)} |w\varphi| = 1$.

Definition

A Whitehead automorphism of $F_r = \langle a_1, \ldots, a_r \rangle$ is an automorphism of the form $F_r \to F_r$, $a_i \mapsto a_i$, $a_j \mapsto a_i^{\eta\epsilon_i} a_j a_i^{\eta\delta_i}$, where $\eta = \pm 1$, $\epsilon_i = 0, -1$, and $\delta_i = 0, 1$. There are $\sim 2r\, 4^{r-1}$ many.

Lemma (Whitehead, 1936)

Let $w \in F_r$. If there exists $\varphi \in \mathrm{Aut}(F_r)$ with $|w\varphi| < |w|$ then there exists a Whitehead automorphism α such that $|w\alpha| < |w|$.

Classical Whitehead's algorithm

Theorem (Whitehead, 1936)

There is an algorithm $\mathcal{W}$ taking $w \in F_r$ as input, deciding whether w is primitive in F_r, and working in time $\mathrm{wc}_{\mathcal{W}}(n) = O(4^r r\, n^2) = O(n^2)$.

Observation

A given $w \in F_r$ is primitive $\Leftrightarrow \min_{\varphi \in \mathrm{Aut}(F_r)} |w\varphi| = 1$.

Definition

A Whitehead automorphism of $F_r = \langle a_1, \ldots, a_r \rangle$ is an automorphism of the form $F_r \to F_r$, $a_i \mapsto a_i$, $a_j \mapsto a_i^{\eta\epsilon_i} a_j a_i^{\eta\delta_i}$, where $\eta = \pm 1$, $\epsilon_i = 0, -1$, and $\delta_i = 0, 1$. There are $\sim 2r\, 4^{r-1}$ many.

Lemma (Whitehead, 1936)

Let $w \in F_r$. If there exists $\varphi \in \mathrm{Aut}(F_r)$ with $|w\varphi| < |w|$ then there exists a Whitehead automorphism α such that $|w\alpha| < |w|$.

Classical Whitehead's algorithm

Theorem (Whitehead, 1936)

There is an algorithm $\mathcal{W}$ taking $w \in F_r$ as input, deciding whether w is primitive in F_r, and working in time $\mathrm{WC}_{\mathcal{W}}(n) = O(4^r r\, n^2) = O(n^2)$.

Observation

A given $w \in F_r$ is primitive $\Leftrightarrow \min_{\varphi \in \mathrm{Aut}(F_r)} |w\varphi| = 1$.

Definition

A Whitehead automorphism of $F_r = \langle a_1, \ldots, a_r \rangle$ is an automorphism of the form $F_r \to F_r$, $a_i \mapsto a_i$, $a_j \mapsto a_i^{\eta\epsilon_i} a_j a_i^{\eta\delta_i}$, where $\eta = \pm 1$, $\epsilon_i = 0, -1$, and $\delta_i = 0, 1$. There are $\sim 2r\, 4^{r-1}$ many.

Lemma (Whitehead, 1936)

Let $w \in F_r$. If there exists $\varphi \in \mathrm{Aut}(F_r)$ with $|w\varphi| < |w|$ then there exists a Whitehead automorphism α such that $|w\alpha| < |w|$.

Classical Whitehead's algorithm

Theorem (Whitehead, 1936)

There is an algorithm $\mathcal{W}$ taking $w \in F_r$ as input, deciding whether w is primitive in F_r, and working in time $\mathrm{wc}_{\mathcal{W}}(n) = O(4^r r\, n^2) = O(n^2)$.

Observation

A given $w \in F_r$ is primitive $\Leftrightarrow \min_{\varphi \in \mathrm{Aut}(F_r)} |w\varphi| = 1$.

Definition

A Whitehead automorphism of $F_r = \langle a_1, \ldots, a_r \rangle$ is an automorphism of the form $F_r \to F_r$, $a_i \mapsto a_i$, $a_j \mapsto a_i^{\eta\epsilon_i} a_j a_i^{\eta\delta_i}$, where $\eta = \pm 1$, $\epsilon_i = 0, -1$, and $\delta_i = 0, 1$. There are $\sim 2r\, 4^{r-1}$ many.

Lemma (Whitehead, 1936)

Let $w \in F_r$. If there exists $\varphi \in \mathrm{Aut}(F_r)$ with $|w\varphi| < |w|$ then there exists a Whitehead automorphism α such that $|w\alpha| < |w|$.

Whitehead cut vertex lemma

Definition

Let $w = a_{i_1}^{\epsilon_1} \cdots a_{i_n}^{\epsilon_n} \in F_r$ be a cyclically reduced word. The Whitehead (unoriented) graph of w, denoted $Wh(w)$, is: $V = \{a_1^{\pm 1}, \ldots, a_r^{\pm 1}\}$ and $E = \{\{a_{i_j}^{\epsilon_j}, a_{i_{j+1}}^{-\epsilon_{j+1}}\} \mid j = 1, \ldots, n \pmod n)\}$.

Theorem (Whitehead's cut vertex lemma)

If $w \in F_r$ is primitive then $Wh(w)$ is either disconnected or has a cut vertex.

Modern proofs/variations given by Heusener–Weidmann and by Wilton.

Proposition (Roig–Weil–V., '07)

Let $w \in F_r$. In view of $Wh(w)$, one can construct (one of the) Whitehead automorphisms decreasing $|w|$ as much as possible, in polynomial time w.r.t. both $n = |w|$ and $r = \mathrm{rk}(F_r)$.

Whitehead cut vertex lemma

Definition

Let $w = a_{i_1}^{\epsilon_1} \cdots a_{i_n}^{\epsilon_n} \in F_r$ be a cyclically reduced word. The Whitehead (unoriented) graph of w, denoted $Wh(w)$, is: $V = \{a_1^{\pm 1}, \ldots, a_r^{\pm 1}\}$ and $E = \{\{a_{i_j}^{\epsilon_j}, a_{i_{j+1}}^{-\epsilon_{j+1}}\} \mid j = 1, \ldots, n \pmod{n}\}$.

Theorem (Whitehead's cut vertex lemma)

If $w \in F_r$ is primitive then $Wh(w)$ is either disconnected or has a cut vertex.

Modern proofs/variations given by Heusener–Weidmann and by Wilton.

Proposition (Roig–Weil–V., '07)

Let $w \in F_r$. In view of $Wh(w)$, one can construct (one of the) Whitehead automorphisms decreasing $|w|$ as much as possible, in polynomial time w.r.t. both $n = |w|$ and $r = \mathrm{rk}(F_r)$.

Whitehead cut vertex lemma

Definition

Let $w = a_{i_1}^{\epsilon_1} \cdots a_{i_n}^{\epsilon_n} \in F_r$ be a cyclically reduced word. The Whitehead (unoriented) graph of w, denoted $Wh(w)$, is: $V = \{a_1^{\pm 1}, \ldots, a_r^{\pm 1}\}$ and $E = \{\{a_{i_j}^{\epsilon_j}, a_{i_{j+1}}^{-\epsilon_{j+1}}\} \mid j = 1, \ldots, n \pmod{n}\}$.

Theorem (Whitehead's cut vertex lemma)

If $w \in F_r$ is primitive then $Wh(w)$ is either disconnected or has a cut vertex.

Modern proofs/variations given by Heusener–Weidmann and by Wilton.

Proposition (Roig–Weil–V., '07)

Let $w \in F_r$. In view of $Wh(w)$, one can construct (one of the) Whitehead automorphisms decreasing $|w|$ as much as possible, in polynomial time w.r.t. both $n = |w|$ and $r = \mathrm{rk}(F_r)$.

Whitehead cut vertex lemma

Definition

Let $w = a_{i_1}^{\epsilon_1} \cdots a_{i_n}^{\epsilon_n} \in F_r$ be a cyclically reduced word. The Whitehead (unoriented) graph of w, denoted $Wh(w)$, is: $V = \{a_1^{\pm 1}, \ldots, a_r^{\pm 1}\}$ and $E = \{\{a_{i_j}^{\epsilon_j}, a_{i_{j+1}}^{-\epsilon_{j+1}}\} \mid j = 1, \ldots, n \pmod{n}\}$.

Theorem (Whitehead's cut vertex lemma)

If $w \in F_r$ is primitive then $Wh(w)$ is either disconnected or has a cut vertex.

Modern proofs/variations given by Heusener–Weidmann and by Wilton.

Proposition (Roig–Weil–V., '07)

Let $w \in F_r$. In view of $Wh(w)$, one can construct (one of the) Whitehead automorphisms decreasing $|w|$ as much as possible, in polynomial time w.r.t. both $n = |w|$ and $r = \mathrm{rk}(F_r)$.

Roig–Weil–V. 's improvement

So, here is a truly polynomial algorithm for checking primitivity:

Algorithm $\mathcal{W}$: • *Given a cyclically reduced $w \in F_r$ with $|w| = n$;*
•[1]: *-If $|w| = 1$, answer* $\mathtt{YES}$ *and* $\mathtt{STOP}$;
-Construct $Wh(w)$ and check whether it is connected and has no cut vertex; if so, answer $\mathtt{NO}$ *and* $\mathtt{STOP}$;
-Otherwise, construct the best possible Whitehead auto φ for w, and repeat Step 1 with $w\varphi$ replacing w.

Theorem (Roig–Weil–V., '07)

The algorithm $\mathcal{W}$ above works in time $\mathrm{wc}_{\mathcal{W}}(n) = O(r^3 n^2)$.

Roig–Weil–V. 's improvement

So, here is a truly polynomial algorithm for checking primitivity:

Algorithm $\mathcal{W}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$;
•[1]: -*If $|w| = 1$, answer* `YES` *and* `STOP`*;*
-*Construct $Wh(w)$ and check whether it is connected and has no cut vertex; if so, answer* `NO` *and* `STOP`*;*
-*Otherwise, construct the best possible Whitehead auto φ for w, and repeat Step 1 with $w\varphi$ replacing w.*

Theorem (Roig–Weil–V., '07)

The algorithm $\mathcal{W}$ above works in time $\mathrm{wc}_{\mathcal{W}}(n) = O(r^3 n^2)$.

Roig–Weil–V. 's improvement

So, here is a truly polynomial algorithm for checking primitivity:

Algorithm $\mathcal{W}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$;
•[1]: *-If $|w| = 1$, answer* $\texttt{YES}$ *and* $\texttt{STOP}$;

-Construct $Wh(w)$ and check whether it is connected and has no cut vertex; if so, answer $\texttt{NO}$ *and* $\texttt{STOP}$;

-Otherwise, construct the best possible Whitehead auto φ for w, and repeat Step 1 with $w\varphi$ replacing w.

Theorem (Roig–Weil–V., '07)

The algorithm $\mathcal{W}$ above works in time $\mathrm{wc}_{\mathcal{W}}(n) = O(r^3 n^2)$.

Roig–Weil–V. 's improvement

So, here is a truly polynomial algorithm for checking primitivity:

Algorithm $\mathcal{W}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$;
•[1]: *-If $|w| = 1$, answer* YES *and* STOP*;*
-Construct $Wh(w)$ and check whether it is connected and has no cut vertex; if so, answer NO *and* STOP*;*
-Otherwise, construct the best possible Whitehead auto φ for w, and repeat Step 1 with $w\varphi$ replacing w.

Theorem (Roig–Weil–V., '07)

The algorithm $\mathcal{W}$ above works in time $\mathrm{wc}_{\mathcal{W}}(n) = O(r^3 n^2)$.

Roig–Weil–V. 's improvement

So, here is a truly polynomial algorithm for checking primitivity:

Algorithm $\mathcal{W}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$;
•[1]: *-If $|w| = 1$, answer* `YES` *and* `STOP`*;*
-Construct $Wh(w)$ and check whether it is connected and has no cut vertex; if so, answer `NO` *and* `STOP`*;*
-Otherwise, construct the best possible Whitehead auto φ for w, and repeat Step 1 with $w\varphi$ replacing w.

Theorem (Roig–Weil–V., '07)

The algorithm $\mathcal{W}$ above works in time $\mathrm{wc}_{\mathcal{W}}(n) = O(r^3 n^2)$.

Roig–Weil–V. 's improvement

So, here is a truly polynomial algorithm for checking primitivity:

Algorithm $\mathcal{W}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$;
•[1]: -If $|w| = 1$, answer `YES` and `STOP`;
-Construct $Wh(w)$ and check whether it is connected and has no cut vertex; if so, answer `NO` and `STOP`;
-Otherwise, construct the best possible Whitehead auto φ for w, and repeat Step 1 with $w\varphi$ replacing w.

Theorem (Roig–Weil–V., '07)

The algorithm $\mathcal{W}$ above works in time $\mathrm{wc}_{\mathcal{W}}(n) = O(r^3 n^2)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer $\texttt{NO}$ and $\texttt{STOP}$;
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; $\texttt{STOP}$.

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • *Given a cyclically reduced $w \in F_r$ with $|w| = n$,*
• *keep constructing $Wh(w)$, edge by edge;*
• *If at some step, the actual graph is connected and has no cut vertex, answer* NO *and* STOP*;*
• *Otherwise, apply $\mathcal{W}$ to decide whether w is primitive;* STOP.

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer `NO` *and* `STOP`*;*
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; `STOP`.

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer $\texttt{NO}$ *and* $\texttt{STOP}$*;*
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; $\texttt{STOP}$.

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time

$$\mathrm{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6).$$

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer $\texttt{NO}$ *and* $\texttt{STOP}$*;*
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; $\texttt{STOP}$.

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathrm{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer `NO` *and* `STOP`*;*
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; `STOP`*.*

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathtt{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathtt{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer NO *and* STOP*;*
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; STOP.

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathtt{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathtt{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Shpilrain's improvement

Shpilrain's idea for fast primitivity checking is as follows:

Algorithm $\mathcal{S}$: • Given a cyclically reduced $w \in F_r$ with $|w| = n$,
• keep constructing $Wh(w)$, edge by edge;
• If at some step, the actual graph is connected and has no cut vertex, answer `NO` *and* `STOP`*;*
• Otherwise, apply $\mathcal{W}$ to decide whether w is primitive; `STOP`*.*

Theorem (Shpilrain, '23)

The above algorithm $\mathcal{S}$ works in time $\mathtt{ac}_{\mathcal{S}}(n) = O(1)$.

... However, this constant depends on the ambient rank r ...

Proposition (Roy–Weil–V.)

Let $r \geqslant 2$. There is $0 < \beta(r) < 1 - \frac{1}{2}r^{-2}$ such that $\mathcal{S}$ works in time $\mathtt{ac}_{\mathcal{S}}(n) = O\left(\left(\frac{r}{1-\beta(r)}\right)^2 + r^3\right) = O(r^6)$.

Outline

Relative Primitivity

Definition (Relative Primitivity Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *(belongs to and) is primitive in* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$.

Definition (Uniform Membership Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *belongs to* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$*; in this case, write* w_0 *in terms of some basis for* H.

We consider the size of the input as $|w_0| + |w_1| + \cdots + |w_k|$*, with ...*

- k *constant:* $I = F_r^{k+1}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=1}^{k} |w_i|$*, or*
- $k \leqslant f(n)$*:* $I = \{(w_0, w_1, \ldots, w_k) \in F_r^{k+1} \mid k \leqslant f(n),\ n = \max_{i=1}^{k} |w_i|\}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=i}^{k} |w_i| \leqslant m + nf(n)$*,*

where $m = |w_0|$.

Relative Primitivity

Definition (Relative Primitivity Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *(belongs to and) is primitive in* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$.

Definition (Uniform Membership Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *belongs to* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$*; in this case, write* w_0 *in terms of some basis for* H.

We consider the size of the input as $|w_0| + |w_1| + \cdots + |w_k|$*, with ...*

- *k constant:* $I = F_r^{k+1}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=1}^{k} |w_i|$*, or*
- *$k \leqslant f(n)$:* $I = \{(w_0, w_1, \ldots, w_k) \in F_r^{k+1} \mid k \leqslant f(n),\ n = \max_{i=1}^{k} |w_i|\}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=i}^{k} |w_i| \leqslant m + nf(n)$,

where $m = |w_0|$.

Relative Primitivity

Definition (Relative Primitivity Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *(belongs to and) is primitive in* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$.

Definition (Uniform Membership Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *belongs to* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$*; in this case, write* w_0 *in terms of some basis for* H.

We consider the size of the input as $|w_0| + |w_1| + \cdots + |w_k|$*, with ...*

- k *constant:* $I = F_r^{k+1}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=1}^{k} |w_i|$*, or*
- $k \leqslant f(n)$*:* $I = \{(w_0, w_1, \ldots, w_k) \in F_r^{k+1} \mid k \leqslant f(n),\ n = \max_{i=1}^{k} |w_i|\}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=i}^{k} |w_i| \leqslant m + nf(n)$*,*

where $m = |w_0|$.

Relative Primitivity

Definition (Relative Primitivity Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *(belongs to and) is primitive in* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$.

Definition (Uniform Membership Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *belongs to* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$*; in this case, write* w_0 *in terms of some basis for* H.

We consider the size of the input as $|w_0| + |w_1| + \cdots + |w_k|$*, with ...*

- k *constant:* $I = F_r^{k+1}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=1}^{k} |w_i|$*, or*
- $k \leqslant f(n)$*:* $I = \{(w_0, w_1, \ldots, w_k) \in F_r^{k+1} \mid k \leqslant f(n), n = \max_{i=1}^{k} |w_i|\}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=i}^{k} |w_i| \leqslant m + nf(n)$,

where $m = |w_0|$.

Relative Primitivity

Definition (Relative Primitivity Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *(belongs to and) is primitive in* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$.

Definition (Uniform Membership Problem)

- *Given* $w_0, w_1, \ldots, w_k \in F_r$;
- *Decide if* w_0 *belongs to* $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$*; in this case, write* w_0 *in terms of some basis for* H.

We consider the size of the input as $|w_0| + |w_1| + \cdots + |w_k|$*, with ...*

- *k constant:* $I = F_r^{k+1}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=1}^{k} |w_i|$*, or*
- $k \leqslant f(n)$: $I = \{(w_0, w_1, \ldots, w_k) \in F_r^{k+1} \mid k \leqslant f(n),\ n = \max_{i=1}^{k} |w_i|\}$ *and* $|(w_0, w_1, \ldots, w_r)| = m + \sum_{i=i}^{k} |w_i| \leqslant m + nf(n)$,

where $m = |w_0|$.

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the Stallings graph $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 spells the label of a closed path at the basepoint of $\Gamma(H)$ answer YES; otherwise answer NO;
• In the affirmative case, construct a maximal tree T in $\Gamma(H)$, construct the corresponding basis B for H, and keep track of the visits of the above closed path to the edges outside T; STOP.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ runs in time $\mathrm{wc}_{\mathcal{MP}}(n) = O(kn\log^*(kn) + m)$, where $n = \max_{i=1,\ldots,k} |w_i|$ and $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the *Stallings graph* $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 *spells the label* of a closed path at the basepoint of $\Gamma(H)$ *answer* YES; otherwise *answer* NO;
• In the affirmative case, *construct a maximal tree* T in $\Gamma(H)$, construct the corresponding *basis* B *for* H, and keep track of the visits of the above closed path to the *edges outside* T; STOP.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ *runs in time* $\mathrm{wc}_{\mathcal{MP}}(n) = O(kn \log^*(kn) + m)$, *where* $n = \max_{i=1,\ldots,k} |w_i|$ *and* $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the Stallings graph $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 spells the label of a closed path at the basepoint of $\Gamma(H)$ answer YES*; otherwise answer* NO*;*
• In the affirmative case, construct a maximal tree T in $\Gamma(H)$, construct the corresponding basis B for H, and keep track of the visits of the above closed path to the edges outside T; STOP.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ runs in time $\mathrm{wc}_{\mathcal{MP}}(n) = O(kn \log^*(kn) + m)$, where $n = \max_{i=1,\ldots,k} |w_i|$ and $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the Stallings graph $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 spells the label of a closed path at the basepoint of $\Gamma(H)$ answer YES*; otherwise answer* NO*;*
• In the affirmative case, construct a maximal tree T in $\Gamma(H)$, construct the corresponding basis B for H, and keep track of the visits of the above closed path to the edges outside T; STOP.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ runs in time $\mathrm{wc}_{\mathcal{MP}}(n) = O(kn\log^*(kn) + m)$, where $n = \max_{i=1,\ldots,k} |w_i|$ and $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the Stallings graph $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 spells the label of a closed path at the basepoint of $\Gamma(H)$ answer `YES`; otherwise answer `NO`;
• In the affirmative case, construct a maximal tree T in $\Gamma(H)$, construct the corresponding basis B for H, and keep track of the visits of the above closed path to the edges outside T; `STOP`.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ runs in time $\mathrm{wc}_{\mathcal{MP}}(n) = O(kn \log^*(kn) + m)$, where $n = \max_{i=1,\ldots,k} |w_i|$ and $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the Stallings graph $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 spells the label of a closed path at the basepoint of $\Gamma(H)$ answer `YES`*; otherwise answer* `NO`*;*
• In the affirmative case, construct a maximal tree T in $\Gamma(H)$, construct the corresponding basis B for H, and keep track of the visits of the above closed path to the edges outside T; `STOP`.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ runs in time $\mathrm{WC}_{\mathcal{MP}}(n) = O(kn\log^*(kn) + m)$, where $n = \max_{i=1,\ldots,k} |w_i|$ and $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

Uniform Membership

Uniform Membership can be nicely solved using Stallings graphs ...

Algorithm $\mathcal{MP}$: • Given $w_0, w_1, \ldots, w_k \in F_r$;
• Construct the Stallings graph $\Gamma(H)$ for $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$;
• If w_0 spells the label of a closed path at the basepoint of $\Gamma(H)$ answer `YES`; otherwise answer `NO`;
• In the affirmative case, construct a maximal tree T in $\Gamma(H)$, construct the corresponding basis B for H, and keep track of the visits of the above closed path to the edges outside T; `STOP`.

Theorem (Touikan, '06)

The algorithm $\mathcal{MP}$ runs in time $\mathrm{WC}_{\mathcal{MP}}(n) = O(kn \log^*(kn) + m)$, where $n = \max_{i=1,\ldots,k} |w_i|$ and $m = |w_0|$.

To solve these problems with low average case complexity, the Central Tree Property will be essential ...

The Central Tree Property

Definition

Let $d \geqslant 1$. We say that the k-tuple $\mathbf{w} = (w_1, \ldots, w_k) \in F_r^k$ has the d-central tree property (d-CTP) if $\min_{i=1}^{k} |w_i| \geqslant 2d+1$, and the $2d$ prefixes of length d of the $w_i^{\pm 1}$'s,

$$w_i = pr_d(w_i) \cdot mf_d(w_i) \cdot pr_d(w_i^{-1})^{-1},$$

are pairwise distinct. We say that $\mathbf{w}$ has the CTP if it has the d-CTP for some $1 \leqslant d < n/2$, where $n = \min_{i=1}^{k} |w_i|$.

Observation

Let $\mathbf{w} = (w_1, \ldots, w_k)$ and $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$. If $\mathbf{w}$ has the CTP then the Stallings graph $\Gamma(H)$ consists on the 'tree of prefixes' plus k arcs connecting their leaves; in particular, $\mathrm{rk}(H) = k$ and $\{w_1, \ldots, w_k\}$ is a free basis for H.

The Central Tree Property

Definition

Let $d \geqslant 1$. We say that the k-tuple $\mathbf{w} = (w_1, \ldots, w_k) \in F_r^k$ has the d-central tree property (d-CTP) if $\min_{i=1}^k |w_i| \geqslant 2d+1$, and the $2d$ prefixes of length d of the $w_i^{\pm 1}$'s,

$$w_i = pr_d(w_i) \cdot mf_d(w_i) \cdot pr_d(w_i^{-1})^{-1},$$

are pairwise distinct. We say that $\mathbf{w}$ has the CTP if it has the d-CTP for some $1 \leqslant d < n/2$, where $n = \min_{i=1}^k |w_i|$.

Observation

Let $\mathbf{w} = (w_1, \ldots, w_k)$ and $H = \langle w_1, \ldots, w_k \rangle \leqslant F_r$. If $\mathbf{w}$ has the CTP then the Stallings graph $\Gamma(H)$ consists on the 'tree of prefixes' plus k arcs connecting their leaves; in particular, $\mathrm{rk}(H) = k$ and $\{w_1, \ldots, w_k\}$ is a free basis for H.

Membership Problem solved fast

Lemma

Let $d(n)$ be a non-decreasing function with $d(n) < n/2$. A random k-tuple of words in F_r of length up to n fails the $d(n)$-CTP with probability $O(k^2(2r-1)^{-d(n/2)})$.

For an increasing function $d(n)$ with $d(n) < n/2$, consider Algorithm $\mathcal{MP}_d$: • Given $w_0, w_1, \ldots, w_k \in F_r$;

•[1] *-Compute $n = \max_{i=1}^{k} |w_i|$.*

-Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;

-If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H; `STOP`*;*

•[2] *$\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;*

-Start reading w_0 in $\Gamma(H)$ from the basepoint, and keeping track of the sequence of arcs fully crossed;

-If the reading cannot be completed to a closed path answer `NO`*; otherwise, answer* `YES` *and output the expression of w_0 in the free basis $\{w_1, \ldots, w_k\}$ for H;* `STOP`*.*

Membership Problem solved fast

Lemma

Let $d(n)$ be a non-decreasing function with $d(n) < n/2$. A random k-tuple of words in F_r of length up to n fails the $d(n)$-CTP with probability $O(k^2(2r-1)^{-d(n/2)})$.

For an increasing function $d(n)$ with $d(n) < n/2$, consider *Algorithm $\mathcal{MP}_d$*: • *Given $w_0, w_1, \ldots, w_k \in F_r$;*

•[1] *-Compute $n = \max_{i=1}^{k} |w_i|$.*

-Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;

-If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H; $\mathtt{STOP}$;

•[2] *$\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;*

-Start reading w_0 in $\Gamma(H)$ from the basepoint, and keeping track of the sequence of arcs fully crossed;

-If the reading cannot be completed to a closed path answer $\mathtt{NO}$; otherwise, answer $\mathtt{YES}$ and output the expression of w_0 in the free basis $\{w_1, \ldots, w_k\}$ for H; $\mathtt{STOP}$.

Membership Problem solved fast

Lemma

Let $d(n)$ be a non-decreasing function with $d(n) < n/2$. A random k-tuple of words in F_r of length up to n fails the $d(n)$-CTP with probability $O(k^2(2r-1)^{-d(n/2)})$.

For an increasing function $d(n)$ with $d(n) < n/2$, consider Algorithm $\mathcal{MP}_d$: • Given $w_0, w_1, \ldots, w_k \in F_r$;

•[1] -*Compute $n = \max_{i=1}^{k} |w_i|$.*

-*Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;*

-*If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H;* `STOP`;

•[2] *$\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;*

-*Start reading w_0 in $\Gamma(H)$ from the basepoint, and keeping track of the sequence of arcs fully crossed;*

-*If the reading cannot be completed to a closed path answer* `NO`; *otherwise, answer* `YES` *and output the expression of w_0 in the free basis $\{w_1, \ldots, w_k\}$ for H;* `STOP`.

Membership Problem solved fast

Lemma

Let $d(n)$ be a non-decreasing function with $d(n) < n/2$. A random k-tuple of words in F_r of length up to n fails the $d(n)$-CTP with probability $O(k^2(2r-1)^{-d(n/2)})$.

For an increasing function $d(n)$ with $d(n) < n/2$, consider Algorithm $\mathcal{MP}_d$: • Given $w_0, w_1, \ldots, w_k \in F_r$;

•[1] *-Compute $n = \max_{i=1}^{k} |w_i|$.*

-Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;

-If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H; `STOP`*;*

•[2] *$\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;*

-Start reading w_0 in $\Gamma(H)$ from the basepoint, and keeping track of the sequence of arcs fully crossed;

-If the reading cannot be completed to a closed path answer `NO`*; otherwise, answer* `YES` *and output the expression of w_0 in the free basis $\{w_1, \ldots, w_k\}$ for H;* `STOP`*.*

Membership Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{MP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If
(i) $k(n)$ is constant then $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O(n \log^* n + m)$;
(ii) $k(n) = n^\beta$, $\beta > 0$, then $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+\gamma} + mn^{2\beta}(2r-1)^{-n^\gamma}\big)$ for any $0 < \gamma < 1$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+1} \log^* n + m\big)$;
(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{18}$ then, for $0 < \epsilon < \frac{1}{8} - \frac{9\beta}{4}$, $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} + m(2r-1)^{(\frac{9}{4}\beta - \frac{1}{8} + \epsilon)n}\big)$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} \log^* n + m\big)$.

And combined with the relative version of algorithm $\mathcal{S}$, we can solve the Relative Primitivity Problem fast:

Membership Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{MP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If
(i) $k(n)$ is constant then $\mathrm{ac}_{\mathcal{MP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$, *while* $\mathrm{wc}_{\mathcal{MP}_d}(n) = O\big(n \log^* n + m\big)$;
(ii) $k(n) = n^{\beta}$, $\beta > 0$, then $\mathrm{ac}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+\gamma} + mn^{2\beta}(2r-1)^{-n^{\gamma}}\big)$ *for any $0 < \gamma < 1$, while* $\mathrm{wc}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+1} \log^* n + m\big)$;
(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{18}$ then, for $0 < \epsilon < \frac{1}{8} - \frac{9\beta}{4}$, $\mathrm{ac}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} + m(2r-1)^{(\frac{9}{4}\beta - \frac{1}{8} + \epsilon)n}\big)$, *while* $\mathrm{wc}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} \log^* n + m\big)$.

And combined with the relative version of algorithm $\mathcal{S}$, we can solve the Relative Primitivity Problem fast:

Membership Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{MP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If
(i) $k(n)$ is constant then $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n \log^* n + m\big)$;
(ii) $k(n) = n^{\beta}$, $\beta > 0$, then $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+\gamma} + mn^{2\beta}(2r-1)^{-n^{\gamma}}\big)$ for any $0 < \gamma < 1$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+1}\log^* n + m\big)$;
(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{18}$ then, for $0 < \epsilon < \frac{1}{8} - \frac{9\beta}{4}$, $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} + m(2r-1)^{(\frac{9}{4}\beta - \frac{1}{8} + \epsilon)n}\big)$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n}\log^* n + m\big)$.

And combined with the relative version of algorithm $\mathcal{S}$, we can solve the Relative Primitivity Problem fast:

Membership Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{MP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If
(i) $k(n)$ is constant then $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n \log^* n + m\big)$;
(ii) $k(n) = n^{\beta}$, $\beta > 0$, then $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+\gamma} + mn^{2\beta}(2r-1)^{-n^{\gamma}}\big)$ for any $0 < \gamma < 1$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n^{\beta+1} \log^* n + m\big)$;
(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{18}$ then, for $0 < \epsilon < \frac{1}{8} - \frac{9\beta}{4}$, $\mathtt{ac}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} + m(2r-1)^{(\frac{9}{4}\beta - \frac{1}{8} + \epsilon)n}\big)$, while $\mathtt{wc}_{\mathcal{MP}_d}(n) = O\big(n(2r-1)^{\beta n} \log^* n + m\big)$.

And combined with the relative version of algorithm $\mathcal{S}$, we can solve the Relative Primitivity Problem fast:

Relative Primitivity Problem solved fast

For an increasing function $d(n)$ with $d(n) < n/2$, consider Algorithm $\mathcal{RP}_d$: • Given $w_0, w_1, \ldots, w_k \in F_r$;

•[1] -Compute $n = \max_{i=1}^{k} |w_i|$.

-Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;

-If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H; then run $\mathcal{S}$ to check whether w_0 is primitive in H; STOP;

•[2] $\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;

-Start reading w_0 in $\Gamma(H)$ from the basepoint, keeping track of the sequence of arcs fully crossed, and constructing the graph $Wh(w)$ (w.r.t. $\{w_1, \ldots, w_k\}$) edge by edge;

-If it cannot be completed to a closed path answer NO; STOP;

-If the actual portion of $Wh(w)$ is connected and has no cut vertex, answer NO; STOP;

-Otherwise, apply $\mathcal{W}$ to check whether the element $w \in H$ is primitive in H; STOP.

Relative Primitivity Problem solved fast

For an increasing function $d(n)$ with $d(n) < n/2$, consider Algorithm $\mathcal{RP}_d$: • Given $w_0, w_1, \ldots, w_k \in F_r$;

•[1] *-Compute $n = \max_{i=1}^{k} |w_i|$.*

-Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;

-If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H; then run $\mathcal{S}$ to check whether w_0 is primitive in H; `STOP`;

•[2] *$\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;*

-Start reading w_0 in $\Gamma(H)$ from the basepoint, keeping track of the sequence of arcs fully crossed, and constructing the graph $Wh(w)$ (w.r.t. $\{w_1, \ldots, w_k\}$) edge by edge;

-If it cannot be completed to a closed path answer `NO`; `STOP`;

-If the actual portion of $Wh(w)$ is connected and has no cut vertex, answer `NO`; `STOP`;

-Otherwise, apply $\mathcal{W}$ to check whether the element $w \in H$ is primitive in H; `STOP`.

Relative Primitivity Problem solved fast

For an increasing function $d(n)$ with $d(n) < n/2$, consider Algorithm $\mathcal{RP}_d$: • Given $w_0, w_1, \ldots, w_k \in F_r$;

•[1] *-Compute $n = \max_{i=1}^{k} |w_i|$.*

-Construct the tree of $d(n)$-prefixes $\Gamma_{d(n)}(\mathbf{w})$;

-If $\mathbf{w}$ has the $d(n)$-CTP and $\min_{i=1}^{k} |w_i| > n/2$ go to Step 2; otherwise, run $\mathcal{MP}$ to decide whether $w_0 \in H$ and find an expression for it in some basis for H; then run $\mathcal{S}$ to check whether w_0 is primitive in H; `STOP`*;*

•[2] *$\Gamma(H)$ equals $\Gamma_{d(n)}(\mathbf{w})$ plus k arcs labeled $mf_{d(n)}(w_i)$;*

-Start reading w_0 in $\Gamma(H)$ from the basepoint, keeping track of the sequence of arcs fully crossed, and constructing the graph $Wh(w)$ (w.r.t. $\{w_1, \ldots, w_k\}$) edge by edge;

-If it cannot be completed to a closed path answer `NO`*;* `STOP`*;*

-If the actual portion of $Wh(w)$ is connected and has no cut vertex, answer `NO`*;* `STOP`*;*

-Otherwise, apply $\mathcal{W}$ to check whether the element $w \in H$ is primitive in H; `STOP`*.*

Relative Primitivity Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{RP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If
(i) $k(n)$ is constant then $\mathrm{ac}_{\mathcal{RP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$;
(ii) $k(n) = n^{\beta}$, $\beta > 0$, then for any $0 < \gamma < 1$,
$\mathrm{ac}_{\mathcal{RP}_d}(n) = O\big(n^{\beta+\gamma} + n^{2\beta}(2r-1)^{-n^{\gamma}}m + n^{6\beta}\big(\frac{2}{2r-1}\big)^{m}\big)$;
(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{58}$ then,
$\mathrm{ac}_{\mathcal{RP}_d}(n) = O\big(n(2r-1)^{\beta n} + (2r-1)^{-5\beta n}m + (2r-1)^{6\beta n - \frac{1-58\beta}{1-56\beta}m}\big)$.

Relative Primitivity Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{RP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If

(i) $k(n)$ is constant then $\mathtt{ac}_{\mathcal{RP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$;

(ii) $k(n) = n^{\beta}$, $\beta > 0$, then for any $0 < \gamma < 1$,

$\mathtt{ac}_{\mathcal{RP}_d}(n) = O\big(n^{\beta+\gamma} + n^{2\beta}(2r-1)^{-n^{\gamma}} m + n^{6\beta}\big(\frac{2}{2r-1}\big)^{m}\big)$;

(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{58}$ then,

$\mathtt{ac}_{\mathcal{RP}_d}(n) = O\big(n(2r-1)^{\beta n} + (2r-1)^{-5\beta n} m + (2r-1)^{6\beta n - \frac{1-58\beta}{1-56\beta} m}\big)$.

Relative Primitivity Problem solved fast

Theorem (Roy–Weil–V.)

Consider the algorithm $\mathcal{RP}_d$ with input a word of length m and a $k(n)$-tuple of words of length at most n in F_r. If

(i) $k(n)$ is constant then $\mathtt{ac}_{\mathcal{RP}_d}(n) = O\big(\log n + mn^{-\log(2r-1)}\big)$;

(ii) $k(n) = n^\beta$, $\beta > 0$, then for any $0 < \gamma < 1$,

$\mathtt{ac}_{\mathcal{RP}_d}(n) = O\big(n^{\beta+\gamma} + n^{2\beta}(2r-1)^{-n^\gamma} m + n^{6\beta}\big(\frac{2}{2r-1}\big)^m\big)$;

(iii) $k(n) = (2r-1)^{\beta n}$, $0 < \beta < \frac{1}{58}$ then,

$\mathtt{ac}_{\mathcal{RP}_d}(n) = O\big(n(2r-1)^{\beta n} + (2r-1)^{-5\beta n} m + (2r-1)^{6\beta n - \frac{1-58\beta}{1-56\beta} m}\big)$.

THANKS