Whitehead minimization in polynomial time

Enric Ventura

Departament de Matemàtica Aplicada III

Universitat Politècnica de Catalunya

Technion, Haifa

Nov. 30th, 2009.

Outline

1. The classical Whitehead algorithm
2. Let's do it in polynomial time
3. The bijection between subgroups and automata
4. Whitehead minimization for subgroups
5. An application

Outline

1. The classical Whitehead algorithm
2. Let's do it in polynomial time
3. The bijection between subgroups and automata
4. Whitehead minimization for subgroups
5. An application

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A: $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically): $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A: $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically): $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A: $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically): $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.

Definitions and notation

- $A = \{a_1, \ldots, a_k\}$ is a finite alphabet (n letters).
- $A^{\pm 1} = A \cup A^{-1} = \{a_1, a_1^{-1}, \ldots, a_k, a_k^{-1}\}$.
- Usually, $A = \{a, b, c\}$.
- $(A^{\pm 1})^*$ the free monoid on $A^{\pm 1}$ (words on $A^{\pm 1}$).
- $F_A = (A^{\pm 1})^* / \sim$ is the free group on A (words on $A^{\pm 1}$ modulo reduction).
- Every $w \in A^*$ has a unique reduced form,
- 1 denotes the empty word, and $|\cdot|$ the (shortest) length in F_A:
  $|1| = 0, \quad |aba^{-1}| = |abbb^{-1}a^{-1}| = 3, \quad |uv| \leqslant |u| + |v|$.
- $\|\cdot\|$ denotes the (shortest) length in the conjugacy class (i.e. cyclically):
  $\|abbb^{-1}a^{-1}\| = 1$.

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":
First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. □

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":
First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. □

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":

First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. □

Whitehead problem

Whitehead Problem

For a group G, find an algorithm s.t. given $u, v \in G$ decides whether there exists $\varphi \in Aut(G)$ such that $\varphi(u) = v$.

Theorem (Whitehead, 30's)

Whitehead problem is solvable in F_A.

"Proof":
First part: reduce $\|u\|$ and $\|v\|$ as much as possible by applying autos:

$$u \to u_1 \to u_2 \to \cdots \to u',$$

$$v \to v_1 \to v_2 \to \cdots \to v'.$$

Second part: analyze who is image of who by some auto, in the (finite!) sphere of given radius n, $S_n = \{w \in F_k \mid \|w\| = n\}$. □

Whitehead minimization problem

Let us concentrate in the first part:

Whitehead Minimization Problem (WMP)

Given $u \in F_A$, find $\varphi \in Aut(F_A)$ such that $\|\varphi(u)\|$ is minimal.

Lemma (Whitehead)

Let $u \in F_A$. If $\exists\, \varphi \in Aut(F_A)$ such that $\|\varphi(u)\| < \|u\|$ then $\exists$ a "Whitehead automorphism" α such that $\|\alpha(u)\| < \|u\|$.

Definition

Whitehead automorphisms are those of the form:

$$\begin{array}{rcll} F_A & \to & F_A & \\ a_i & \mapsto & a_i & \text{(the multiplier)} \\ a_i \neq a_j & \mapsto & a_i^{\epsilon_j} a_j a_i^{\delta_j} & \end{array}$$

where $\epsilon_j = 0, -1$ and $\delta_j = 0, 1$ (there are $\sim k \cdot 4^k$ many, where $k = |A|$).

Whitehead minimization problem

Let us concentrate in the first part:

Whitehead Minimization Problem (WMP)

Given $u \in F_A$, find $\varphi \in Aut(F_A)$ such that $\|\varphi(u)\|$ is minimal.

Lemma (Whitehead)

Let $u \in F_A$. If $\exists\, \varphi \in Aut(F_A)$ such that $\|\varphi(u)\| < \|u\|$ then $\exists$ a "Whitehead automorphism" α such that $\|\alpha(u)\| < \|u\|$.

Definition

Whitehead automorphisms are those of the form:

$$\begin{array}{rcl} F_A & \to & F_A \\ a_i & \mapsto & a_i \qquad \text{(the multiplier)} \\ a_i \neq a_j & \mapsto & a_i^{\epsilon_j} a_j a_i^{\delta_j} \end{array}$$

where $\epsilon_j = 0, -1$ and $\delta_j = 0, 1$ (there are $\sim k \cdot 4^k$ many, where $k = |A|$).

Whitehead minimization problem

Let us concentrate in the first part:

Whitehead Minimization Problem (WMP)

Given $u \in F_A$, find $\varphi \in Aut(F_A)$ such that $\|\varphi(u)\|$ is minimal.

Lemma (Whitehead)

Let $u \in F_A$. If $\exists\, \varphi \in Aut(F_A)$ such that $\|\varphi(u)\| < \|u\|$ then $\exists$ a "Whitehead automorphism" α such that $\|\alpha(u)\| < \|u\|$.

Definition

Whitehead automorphisms are those of the form:

$$\begin{array}{rcl} F_A & \to & F_A \\ a_i & \mapsto & a_i \qquad \text{(the multiplier)} \\ a_i \neq a_j & \mapsto & a_i^{\epsilon_j} a_j a_i^{\delta_j} \end{array}$$

where $\epsilon_j = 0, -1$ and $\delta_j = 0, 1$ (there are $\sim k \cdot 4^k$ many, where $k = |A|$).

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Classical Whitehead's algorithm (first part)

Classical whitehead algorithm is:

- *Keep applying whitehead automorphisms to given u until finding one that decreases its cyclic length.*
- *Repeat until all whiteheads are non-decreasing.*

This is polynomial on $\|u\|$, but exponential on the ambient rank, k.

There are several recent results (theoretical, heuristic, probabilistic) suggesting that Whitehead algorithm is faster in practice.

Outline

1. The classical Whitehead algorithm
2. Let's do it in polynomial time
3. The bijection between subgroups and automata
4. Whitehead minimization for subgroups
5. An application

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2\, k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2\, k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Improvement

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

main idea: given $u \in F_k$, we find in polynomial time one of the whiteheads that decreases $\|u\|$ the most possible.

Key point: How does a given Whitehead automorphism α affect the length of a given word u ?

Three ingredients:

1) Codify u as its Whitehead's graph (classic in Group Theory),
2) Codify α as a cut in this graph ($\approx$ classic in Group Theory),
3) Use max-flow min-cut algorithm (classic in Computer Science),
4) ... put together and mix (new!).

Whitehead's graph

First ingredient: Whitehead's graph of a word.

Definition

Given $u \in F_k$ (cyclically reduced), its (unoriented) Whitehead graph, denoted $Wh(u)$, is:

- *vertices: $A^{\pm 1}$,*
- *edges: for every pair of (cycl.) consecutive letters $u = \cdots xy \cdots$ put an edge between x and y^{-1}.*

Example

$u = aba^{-1}c^{-1}bbabc^{-1}$,

Whitehead's graph

First ingredient: Whitehead's graph of a word.

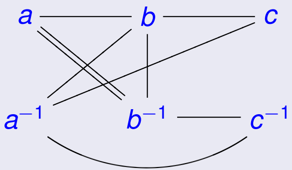
Definition

Given $u \in F_k$ (cyclically reduced), its (unoriented) Whitehead graph, denoted $Wh(u)$, is:

- *vertices: $A^{\pm 1}$,*
- *edges: for every pair of (cycl.) consecutive letters $u = \cdots xy \cdots$ put an edge between x and y^{-1}.*

Example

$$u = aba^{-1}c^{-1}bbabc^{-1},$$



Cut in a graph

Second ingredient: Cut in a graph.

Definition

Given a Whitehead's automorphism α, we represent it as the (a, a^{-1})-cut

$$(T = \{a\} \cup \{\textit{letters that go multiplied on the right by } a\}, a)$$

of the set $A^{\pm 1}$.

Example

$$\begin{array}{rcl} \alpha\colon \langle a, b, c\rangle = F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$$

a	b	c
a^{-1}	b^{-1}	c^{-1}

Cut in a graph

Second ingredient: Cut in a graph.

Definition

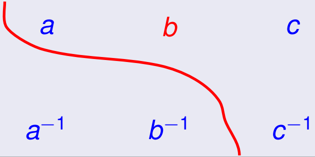
Given a Whitehead's automorphism α, we represent it as the (a, a^{-1})-cut

$$(T = \{a\} \cup \{\textit{letters that go multiplied on the right by } a\}, a)$$

of the set $A^{\pm 1}$.

Example

$$\begin{array}{rcl} \alpha\colon \langle a, b, c\rangle = F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$$



Rephrasement of Wh. Lemma

Lemma (Whitehead)

Given a word $u \in F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(u)$, say $\alpha = (T, a)$, and then

$$\|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(a).$$

Proof: Analyzing combinatorial cases (see Lyndon-Schupp).

Rephrasement of Wh. Lemma

Lemma (Whitehead)

Given a word $u \in F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(u)$, say $\alpha = (T, a)$, and then

$$\|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(a).$$

Proof: Analyzing combinatorial cases (see Lyndon-Schupp).

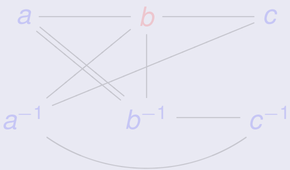
Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and* $\alpha\colon F_3 \to F_3$ *like before. We*

$$\begin{array}{rcl} \alpha\colon F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$$

have $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



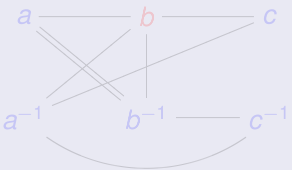
and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and* $\alpha\colon \begin{array}{ccc} F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$ *like before. We have* $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



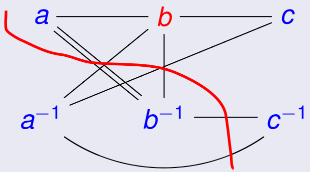
and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and* $\alpha\colon \begin{array}{ccc} F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$ *like before. We have* $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



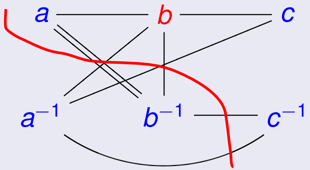
and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Example

Example

Consider $u = aba^{-1}c^{-1}bbabc^{-1}$ *and* $\alpha\colon \begin{array}{ccc} F_3 & \to & F_3 \\ a & \mapsto & ab \\ b & \mapsto & b \\ c & \mapsto & b^{-1}cb \end{array}$ *like before. We have* $\alpha(u) = aba^{-1}b^{-1}c^{-1}bbbabc^{-1}b$. *Furthermore,*



and, in fact,

$$12 - 9 = \|\alpha(u)\| - \|u\| = \mathrm{cap}(T) - \deg(b) = 7 - 4.$$

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Max-flow min-cut algorithm

Third ingredient: Max-flow min-cut algorithm.

Hence, Whitehead's Minimization Problem reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity.*

This can be done by using the classical max-flow min-cut algorithm ...

...which works in polynomial time w.r.t. the number of edges of the graph ($= \|u\|$) and the number of vertices ($= 2k$).

Outline

1 The classical Whitehead algorithm

2 Let's do it in polynomial time

3 The bijection between subgroups and automata

4 Whitehead minimization for subgroups

5 An application

Stallings automata

Definition

A Stallings automaton is a finite A-labeled oriented graph with a distinguished vertex, (X, v)*, such that:*

1- X *is connected,*

2- *no vertex of degree 1 except possibly* v *(*X *is a core-graph),*

3- *no two edges with the same label go out of (or in to) the same vertex.*

NO :

YES :

Stallings automata

Definition

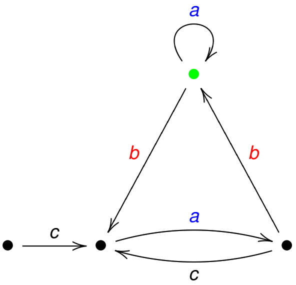
A Stallings automaton *is a finite A-labeled oriented graph with a distinguished vertex,* (X, v)*, such that:*

1- X *is connected,*

2- *no vertex of degree 1 except possibly* v *(*X *is a* core-graph*),*

3- *no two edges with the same label go out of (or in to) the same vertex.*

NO :



YES :

Stallings automata

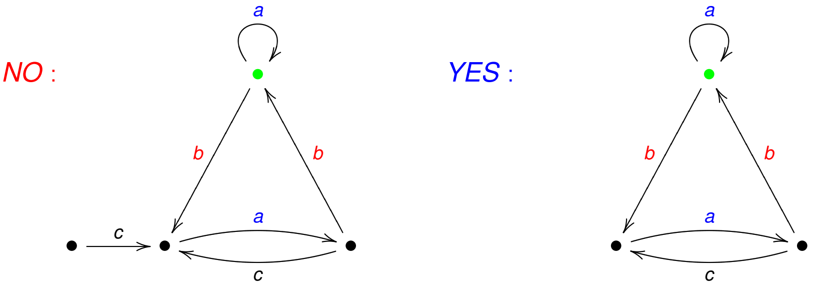
Definition

A Stallings automaton is a finite A-labeled oriented graph with a distinguished vertex, (X, v), such that:

1- *X is connected,*

2- *no vertex of degree 1 except possibly v (X is a core-graph),*

3- *no two edges with the same label go out of (or in to) the same vertex.*



Stallings automata

In the influent paper

J. R. Stallings, Topology of finite graphs, Inventiones Math. 71 (1983), 551-565,

Stallings (building on previous works) gave a bijection between finitely generated subgroups of F_A and Stallings automata:

$$\{\text{f.g. subgroups of } F_A\} \quad \longleftrightarrow \quad \{\text{Stallings automata}\},$$

which is crucial for the modern understanding of the lattice of subgroups of F_A.

Stallings automata

In the influent paper

J. R. Stallings, Topology of finite graphs, Inventiones Math. 71 (1983), 551-565,

Stallings (building on previous works) gave a bijection between finitely generated subgroups of F_A and Stallings automata:

$$\{\text{f.g. subgroups of } F_A\} \quad \longleftrightarrow \quad \{\text{Stallings automata}\},$$

which is crucial for the modern understanding of the lattice of subgroups of F_A.

In the influent paper

J. R. Stallings, Topology of finite graphs, Inventiones Math. 71 (1983), 551-565,

Stallings (building on previous works) gave a bijection between finitely generated subgroups of F_A and Stallings automata:

$$\{\text{f.g. subgroups of } F_A\} \quad \longleftrightarrow \quad \{\text{Stallings automata}\},$$

which is crucial for the modern understanding of the lattice of subgroups of F_A.

Reading the subgroup from the automata

Definition

To any given (Stallings) automaton (X, v), we associate its fundamental group:

$$\pi(X, v) = \{ \text{labels of closed paths at } v \} \leqslant F_A,$$

clearly, a subgroup of F_A.

$X=$

$\pi(X, \bullet) = \{1, a, a^{-1}, bab, bc^{-1}b, babab^{-1}cb^{-1}, \ldots\}$

$\pi(X, \bullet) \not\ni bc^{-1}bcaa$

Membership problem in $\pi(X, \bullet)$ is solvable.

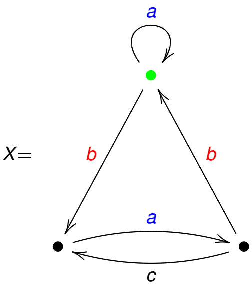
Reading the subgroup from the automata

Definition

To any given (Stallings) automaton (X, v), we associate its fundamental group:

$$\pi(X, v) = \{ \text{ labels of closed paths at } v\} \leqslant F_A,$$

clearly, a subgroup of F_A.



$\pi(X, \bullet) = \{1, a, a^{-1}, bab, bc^{-1}b, babab^{-1}cb^{-1}, \ldots\}$

$\pi(X, \bullet) \not\ni bc^{-1}bcaa$

Membership problem in $\pi(X, \bullet)$ is solvable.

A basis for $\pi(X, v)$

Proposition

For every Stallings automaton (X, v), the group $\pi(X, v)$ is free of rank $rk(\pi(X, v)) = 1 - |VX| + |EX|$.

Proof:

- Take a maximal tree T in X.
- Write $T[p, q]$ for the geodesic (i.e. the unique reduced path) in T from p to q.
- For every $e \in EX - ET$, $x_e = label(T[v, \iota e] \cdot e \cdot T[\tau e, v])$ belongs to $\pi(X, v)$.
- Not difficult to see that $\{x_e \mid e \in EX - ET\}$ is a basis for $\pi(X, v)$.
- And, $$|EX - ET| = |EX| - |ET| = |EX| - (|VT| - 1) = 1 - |VX| + |EX|. \square$$

A basis for $\pi(X, v)$

Proposition

For every Stallings automaton (X, v), the group $\pi(X, v)$ is free of rank $rk(\pi(X, v)) = 1 - |VX| + |EX|$.

Proof:

- Take a maximal tree T in X.
- Write $T[p, q]$ for the geodesic (i.e. the unique reduced path) in T from p to q.
- For every $e \in EX - ET$, $x_e = label(T[v, \iota e] \cdot e \cdot T[\tau e, v])$ belongs to $\pi(X, v)$.
- Not difficult to see that $\{x_e \mid e \in EX - ET\}$ is a basis for $\pi(X, v)$.
- And, $$|EX - ET| = |EX| - |ET| = |EX| - (|VT| - 1) = 1 - |VX| + |EX|. \square$$

A basis for $\pi(X, v)$

Proposition

For every Stallings automaton (X, v)*, the group* $\pi(X, v)$ *is free of rank* $rk(\pi(X, v)) = 1 - |VX| + |EX|$*.*

Proof:

- Take a maximal tree T in X.
- Write $T[p, q]$ for the geodesic (i.e. the unique reduced path) in T from p to q.
- For every $e \in EX - ET$, $x_e = label(T[v, \iota e] \cdot e \cdot T[\tau e, v])$ belongs to $\pi(X, v)$.
- Not difficult to see that $\{x_e \mid e \in EX - ET\}$ is a basis for $\pi(X, v)$.
- And, $$\begin{aligned} |EX - ET| &= |EX| - |ET| \\ &= |EX| - (|VT| - 1) = 1 - |VX| + |EX|. \ \square \end{aligned}$$

A basis for $\pi(X, v)$

Proposition

For every Stallings automaton (X, v), the group $\pi(X, v)$ is free of rank $rk(\pi(X, v)) = 1 - |VX| + |EX|$.

Proof:

- Take a maximal tree T in X.
- Write $T[p, q]$ for the geodesic (i.e. the unique reduced path) in T from p to q.
- For every $e \in EX - ET$, $x_e = label(T[v, \iota e] \cdot e \cdot T[\tau e, v])$ belongs to $\pi(X, v)$.
- Not difficult to see that $\{x_e \mid e \in EX - ET\}$ is a basis for $\pi(X, v)$.
- And, $|EX - ET| = |EX| - |ET| = |EX| - (|VT| - 1) = 1 - |VX| + |EX|$. □

A basis for $\pi(X, v)$

Proposition

For every Stallings automaton (X, v), the group $\pi(X, v)$ is free of rank $rk(\pi(X, v)) = 1 - |VX| + |EX|$.

Proof:

- Take a maximal tree T in X.
- Write $T[p, q]$ for the geodesic (i.e. the unique reduced path) in T from p to q.
- For every $e \in EX - ET$, $x_e = label(T[v, \iota e] \cdot e \cdot T[\tau e, v])$ belongs to $\pi(X, v)$.
- Not difficult to see that $\{x_e \mid e \in EX - ET\}$ is a basis for $\pi(X, v)$.
- And, $$\begin{aligned} |EX - ET| &= |EX| - |ET| \\ &= |EX| - (|VT| - 1) = 1 - |VX| + |EX|. \ \square \end{aligned}$$

A basis for $\pi(X, v)$

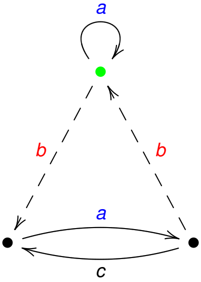
Proposition

For every Stallings automaton (X, v), the group $\pi(X, v)$ is free of rank $rk(\pi(X, v)) = 1 - |VX| + |EX|$.

Proof:

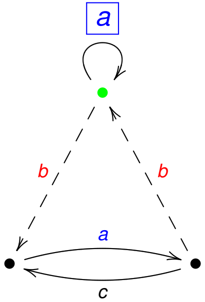
- Take a maximal tree T in X.
- Write $T[p, q]$ for the geodesic (i.e. the unique reduced path) in T from p to q.
- For every $e \in EX - ET$, $x_e = label(T[v, \iota e] \cdot e \cdot T[\tau e, v])$ belongs to $\pi(X, v)$.
- Not difficult to see that $\{x_e \mid e \in EX - ET\}$ is a basis for $\pi(X, v)$.
- And, $$\begin{aligned} |EX - ET| &= |EX| - |ET| \\ &= |EX| - (|VT| - 1) = 1 - |VX| + |EX|. \ \square \end{aligned}$$

Example



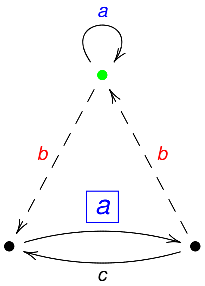
$H = \langle \quad \rangle$

Example



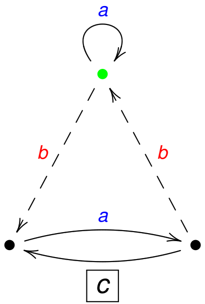
$H = \langle a, \quad \rangle$

Example



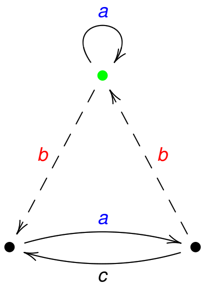
$H = \langle a, \ bab, \quad \rangle$

Example



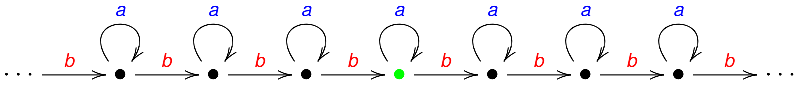
$H = \langle a,\ bab,\ b^{-1}cb^{-1} \rangle$

Example



$H = \langle a,\ bab,\ b^{-1}cb^{-1} \rangle$
$rk(H) = 1 - 3 + 5 = 3.$

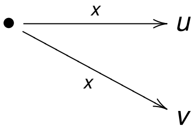
Example-2



$$F_{\aleph_0} \simeq H = \langle \ldots, b^{-2}ab^{2}, b^{-1}ab, a, bab^{-1}, b^{2}ab^{-2}, \ldots \rangle \leqslant F_2.$$

Constructing the automata from the subgroup

In any automaton containing the following situation, for $x \in A^{\pm 1}$,

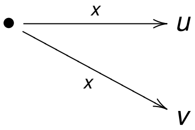


we can fold and identify vertices u and v to obtain

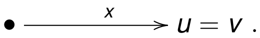
This operation, $(X, v) \rightsquigarrow (X', v)$, is called a Stallings folding.

Constructing the automata from the subgroup

In any automaton containing the following situation, for $x \in A^{\pm 1}$,



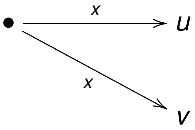
we can fold and identify vertices u and v to obtain



This operation, $(X, v) \rightsquigarrow (X', v)$, is called a Stallings folding.

Constructing the automata from the subgroup

In any automaton containing the following situation, for $x \in A^{\pm 1}$,



we can fold and identify vertices u and v to obtain

$$\bullet \xrightarrow{\quad x \quad} u = v \ .$$

This operation, $(X, v) \rightsquigarrow (X', v)$, is called a Stallings folding.

Constructing the automata from the subgroup

Lemma (Stallings)

If $(X, v) \rightsquigarrow (X', v')$ is a Stallings folding then $\pi(X, v) = \pi(X', v')$.

Given a f.g. subgroup $H = \langle w_1, \ldots w_m \rangle \leqslant F_A$ (we assume w_i are reduced words), do the following:

1- *Draw the flower automaton,*

2- *Perform successive foldings until obtaining a Stallings automaton, denoted $\Gamma(H)$.*

Constructing the automata from the subgroup

Lemma (Stallings)

If $(X, v) \rightsquigarrow (X', v')$ *is a Stallings folding then* $\pi(X, v) = \pi(X', v')$.

Given a f.g. subgroup $H = \langle w_1, \ldots w_m \rangle \leqslant F_A$ *(we assume* w_i *are reduced words), do the following:*

1- *Draw the flower automaton,*

2- *Perform successive foldings until obtaining a Stallings automaton, denoted* $\Gamma(H)$.

Constructing the automata from the subgroup

Lemma (Stallings)

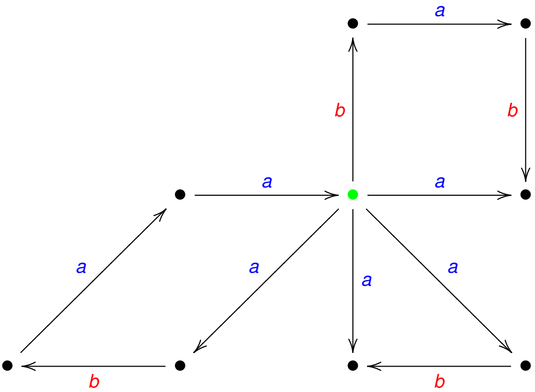
If $(X, v) \rightsquigarrow (X', v')$ is a Stallings folding then $\pi(X, v) = \pi(X', v')$.

Given a f.g. subgroup $H = \langle w_1, \ldots w_m \rangle \leqslant F_A$ (we assume w_i are reduced words), do the following:

1- *Draw the flower automaton,*

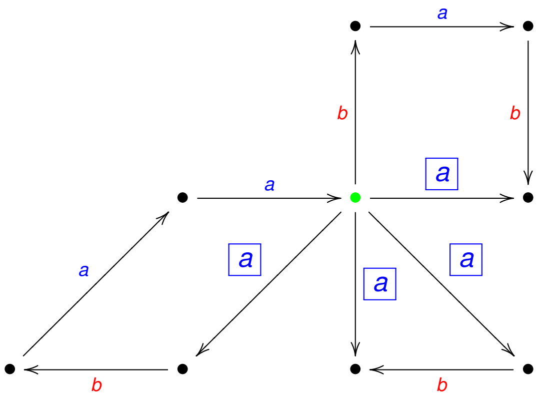
2- *Perform successive foldings until obtaining a Stallings automaton, denoted $\Gamma(H)$.*

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



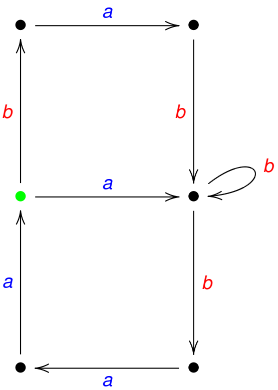
Flower(H)

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



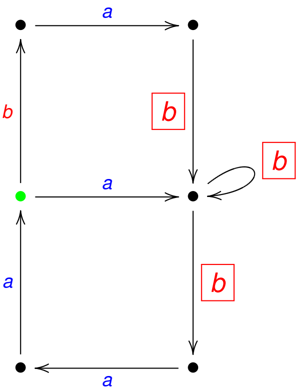
Flower(H)

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



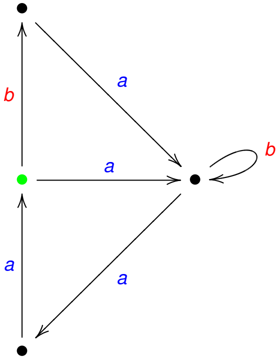
Folding #1

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



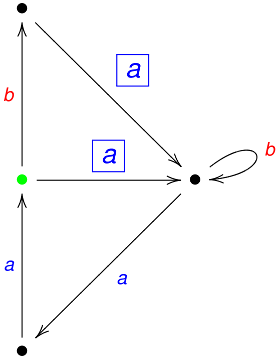
Folding $\#1$.

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



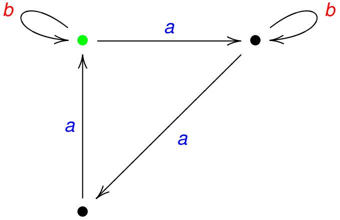
Folding #2.

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



Folding #2.

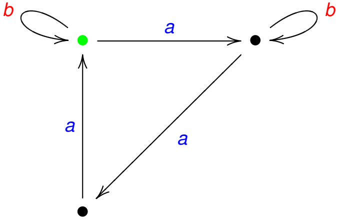
Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



Folding $\#3$. $\Gamma(H)$

By Stallings Lemma, $\pi(\Gamma(H), \bullet) = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$

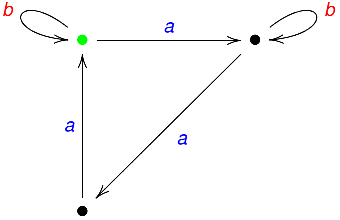
Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



Folding $\#3$. $\Gamma(H)$

By Stallings Lemma, $\pi(\Gamma(H), \bullet) \quad = \quad \langle baba^{-1}, aba^{-1}, aba^2 \rangle$

Example: $H = \langle baba^{-1}, aba^{-1}, aba^2 \rangle$



Folding #3. $\Gamma(H)$

By Stallings Lemma, $\pi(\Gamma(H), \bullet) = \langle baba^{-1}, aba^{-1}, aba^2 \rangle = \langle b, aba^{-1}, a^3 \rangle$

Local confluence

It can be shown that

Proposition

The automaton $\Gamma(H)$ does not depend on the sequence of foldings

Proposition

The automaton $\Gamma(H)$ does not depend on the generators of H.

Theorem

The following is a bijection:

$$\begin{array}{rcl} \{\text{f.g. subgroups of } F_A\} & \longleftrightarrow & \{\text{Stallings automata}\} \\ H & \to & \Gamma(H) \\ \pi(X, v) & \leftarrow & (X, v) \end{array}$$

Local confluence

It can be shown that

Proposition

The automaton $\Gamma(H)$ does not depend on the sequence of foldings

Proposition

The automaton $\Gamma(H)$ does not depend on the generators of H.

Theorem

The following is a bijection:

$$\begin{array}{ccc} \{\text{f.g. subgroups of } F_A\} & \longleftrightarrow & \{\text{Stallings automata}\} \\ H & \to & \Gamma(H) \\ \pi(X, v) & \leftarrow & (X, v) \end{array}$$

Local confluence

It can be shown that

Proposition

The automaton $\Gamma(H)$ does not depend on the sequence of foldings

Proposition

The automaton $\Gamma(H)$ does not depend on the generators of H.

Theorem

The following is a bijection:

$$\begin{array}{rcl} \{\textit{f.g. subgroups of } F_A\} & \longleftrightarrow & \{\textit{Stallings automata}\} \\ H & \to & \Gamma(H) \\ \pi(X, v) & \leftarrow & (X, v) \end{array}$$

Nielsen-Schreier Theorem

Corollary (Nielsen-Schreier)

Every subgroup of F_A is free.

- Finite automata work for the finitely generated case, but everything extends easily to the general case (using infinite graphs).
- The original proof (1920's) is combinatorial and much more technical.

Nielsen-Schreier Theorem

Corollary (Nielsen-Schreier)

Every subgroup of F_A is free.

- Finite automata work for the finitely generated case, but everything extends easily to the general case (using infinite graphs).
- The original proof (1920's) is combinatorial and much more technical.

Nielsen-Schreier Theorem

Corollary (Nielsen-Schreier)

Every subgroup of F_A is free.

- Finite automata work for the finitely generated case, but everything extends easily to the general case (using infinite graphs).
- The original proof (1920's) is combinatorial and much more technical.

Outline

1. The classical Whitehead algorithm
2. Let's do it in polynomial time
3. The bijection between subgroups and automata
4. Whitehead minimization for subgroups
5. An application

Whitehead's hypergraph

A cyclically reduced word can be thought as a circular graph; and then, its Whitehead graph $Wh(u)$ just describes the in-links of the vertices.

Definition

Let $H \leqslant F_k$ be a f.g. subgroup, and let $\Gamma(H)$ be its core graph. We define the Whitehead hyper-graph of H, denoted $Wh(H)$, as:

- *vertices: $A^{\pm 1}$,*
- *hyper-edges: for every vertex v in $\Gamma(H)$, put a hyper-edge consisting on the in-link of v.*

Lemma (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(H)$, say $\alpha = (T, a)$, and then

$$\|\alpha(H)\| - \|H\| = \operatorname{cap}(T) - \deg(a),$$

where $\|H\|$ is the number of vertices in $\Gamma(H)$.

Whitehead's hypergraph

A cyclically reduced word can be thought as a circular graph; and then, its Whitehead graph $Wh(u)$ just describes the in-links of the vertices.

Definition

Let $H \leqslant F_k$ be a f.g. subgroup, and let $\Gamma(H)$ be its core graph. We define the Whitehead hyper-graph of H, denoted $Wh(H)$, as:

- *vertices: $A^{\pm 1}$,*
- *hyper-edges: for every vertex v in $\Gamma(H)$, put a hyper-edge consisting on the in-link of v.*

Lemma (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(H)$, say $\alpha = (T, a)$, and then

$$\|\alpha(H)\| - \|H\| = \operatorname{cap}(T) - \deg(a),$$

where $\|H\|$ is the number of vertices in $\Gamma(H)$.

Whitehead's hypergraph

A cyclically reduced word can be thought as a circular graph; and then, its Whitehead graph $Wh(u)$ just describes the in-links of the vertices.

Definition

Let $H \leqslant F_k$ be a f.g. subgroup, and let $\Gamma(H)$ be its core graph. We define the Whitehead hyper-graph of H, denoted $Wh(H)$, as:

- *vertices: $A^{\pm 1}$,*
- *hyper-edges: for every vertex v in $\Gamma(H)$, put a hyper-edge consisting on the in-link of v.*

Lemma (Roig, V., Weil, 2007)

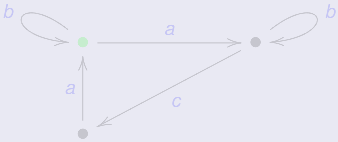
Given a f.g. subgroup $H \leqslant F_k$ and a Whitehead automorphism α, think α as a cut in $Wh(H)$, say $\alpha = (T, a)$, and then

$$\|\alpha(H)\| - \|H\| = \mathrm{cap}(T) - \deg(a),$$

where $\|H\|$ is the number of vertices in $\Gamma(H)$.

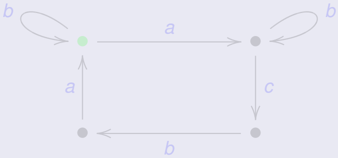
Whitehead's hypergraph

Consider $H = \langle b, aba^{-1}, aca \rangle \leqslant F_3$. *Its core graph* $\Gamma(H)$*, and Whitehead hyper-graph* $Wh(H)$ *are:*



In fact, $\alpha(H) = \langle b, aba^{-1}, acbab \rangle$ *and then*

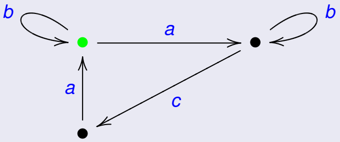
$\Gamma(\alpha(H)) =$

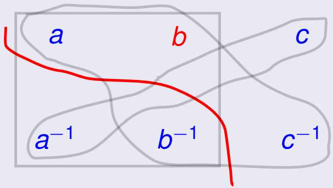


and so, $4 - 3 = \|\alpha(H)\| - \|H\| = 3 - 2$.

Whitehead's hypergraph

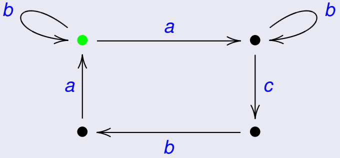
Consider $H = \langle b, aba^{-1}, aca \rangle \leqslant F_3$. *Its core graph* $\Gamma(H)$, *and Whitehead hyper-graph* $Wh(H)$ *are:*





In fact, $\alpha(H) = \langle b, aba^{-1}, acbab \rangle$ *and then*

$\Gamma(\alpha(H)) =$



and so, $4 - 3 = \|\alpha(H)\| - \|H\| = 3 - 2$.

Minimizing capacities in hyper-graphs

So, Whitehead's Minimization Problem for subgroups reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity in the given hyper-graph.*

Unfortunately, there is no analog of max-flow min-cut algorithm for hyper-graphs ...

...but it is still possible to find minimal cuts in polynomial time because of sub-modularity:

Observation

For every f.g. $H \leqslant F_k$, let $W = Wh(H)$ and then the map $\mathcal{P}(A^{\pm 1}) \to \mathbb{N}$, $T \mapsto \mathrm{cap}_W(T)$ is sub-modular.

Minimizing capacities in hyper-graphs

So, Whitehead's Minimization Problem for subgroups reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity in the given hyper-graph.*

Unfortunately, there is no analog of max-flow min-cut algorithm for hyper-graphs ...

...but it is still possible to find minimal cuts in polynomial time because of sub-modularity:

Observation

For every f.g. $H \leqslant F_k$, let $W = Wh(H)$ and then the map $\mathcal{P}(A^{\pm 1}) \to \mathbb{N}$, $T \mapsto \mathrm{cap}_W(T)$ is sub-modular.

Minimizing capacities in hyper-graphs

So, Whitehead's Minimization Problem for subgroups reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity in the given hyper-graph.*

Unfortunately, there is no analog of max-flow min-cut algorithm for hyper-graphs ...

...but it is still possible to find minimal cuts in polynomial time because of sub-modularity:

Observation

For every f.g. $H \leqslant F_k$, let $W = Wh(H)$ and then the map $\mathcal{P}(A^{\pm 1}) \to \mathbb{N}$, $T \mapsto \mathrm{cap}_W(T)$ is sub-modular.

Minimizing capacities in hyper-graphs

So, Whitehead's Minimization Problem for subgroups reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity in the given hyper-graph.*

Unfortunately, there is no analog of max-flow min-cut algorithm for hyper-graphs ...

...but it is still possible to find minimal cuts in polynomial time because of sub-modularity:

Observation

For every f.g. $H \leqslant F_k$, let $W = Wh(H)$ and then the map $\mathcal{P}(A^{\pm 1}) \to \mathbb{N}$, $T \mapsto \mathrm{cap}_W(T)$ is sub-modular.

Minimizing capacities in hyper-graphs

So, Whitehead's Minimization Problem for subgroups reduces to:

- *run over all possible multipliers, say a, (there are $2k$),*
- *find an (a, a^{-1})-cut with minimal possible capacity in the given hyper-graph.*

Unfortunately, there is no analog of max-flow min-cut algorithm for hyper-graphs ...

...but it is still possible to find minimal cuts in polynomial time because of sub-modularity:

Observation

For every f.g. $H \leqslant F_k$, let $W = Wh(H)$ and then the map $\mathcal{P}(A^{\pm 1}) \to \mathbb{N}$, $T \mapsto \mathrm{cap}_W(T)$ is sub-modular.

Sub-modularity

Definition

A map $f\colon \mathcal{P}(V) \to \mathbb{N}$ is called sub-modular *if, for every $A, B \subseteq V$, $f(A \cup B) + f(A \cap B) \leqslant f(A) + f(B)$.*

Efficient minimization of sub-modular functions is an active research topic in computer science. One of the classical results is the following

Proposition

There exists a algorithm which, given a sub-modular function $f\colon \mathcal{P}(V) \to \mathbb{N}$ computes its minimum with a number of queries to evaluate f bounded above by a polynomial on $|V|$.

Corollary

There is an algorithm which solves Whitehead Minimization Problem for subgroups $H \leqslant F_k$, in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Sub-modularity

Definition

A map $f\colon \mathcal{P}(V) \to \mathbb{N}$ *is called* sub-modular *if, for every* $A, B \subseteq V$*,*
$f(A \cup B) + f(A \cap B) \leqslant f(A) + f(B)$.

Efficient minimization of sub-modular functions is an active research topic in computer science. One of the classical results is the following

Proposition

There exists a algorithm which, given a sub-modular function $f\colon \mathcal{P}(V) \to \mathbb{N}$ *computes its minimum with a number of queries to evaluate* f *bounded above by a polynomial on* $|V|$.

Corollary

There is an algorithm which solves Whitehead Minimization Problem for subgroups $H \leqslant F_k$*, in time* $O((n^2k^4 + n^3k^2)\log(nk))$*, where* $n = \|H\|$.

Sub-modularity

Definition

A map $f\colon \mathcal{P}(V) \to \mathbb{N}$ is called sub-modular if, for every $A, B \subseteq V$, $f(A \cup B) + f(A \cap B) \leqslant f(A) + f(B)$.

Efficient minimization of sub-modular functions is an active research topic in computer science. One of the classical results is the following

Proposition

There exists a algorithm which, given a sub-modular function $f\colon \mathcal{P}(V) \to \mathbb{N}$ computes its minimum with a number of queries to evaluate f bounded above by a polynomial on $|V|$.

Corollary

There is an algorithm which solves Whitehead Minimization Problem for subgroups $H \leqslant F_k$, in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Sub-modularity

Definition

A map $f\colon \mathcal{P}(V) \to \mathbb{N}$ is called sub-modular if, for every $A, B \subseteq V$, $f(A \cup B) + f(A \cap B) \leqslant f(A) + f(B)$.

Efficient minimization of sub-modular functions is an active research topic in computer science. One of the classical results is the following

Proposition

There exists a algorithm which, given a sub-modular function $f\colon \mathcal{P}(V) \to \mathbb{N}$ computes its minimum with a number of queries to evaluate f bounded above by a polynomial on $|V|$.

Corollary

There is an algorithm which solves Whitehead Minimization Problem for subgroups $H \leqslant F_k$, in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Outline

1. The classical Whitehead algorithm
2. Let's do it in polynomial time
3. The bijection between subgroups and automata
4. Whitehead minimization for subgroups
5. An application

Primitivity

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Observation

u is primitive $\Leftrightarrow$ the orbit of u contains a.

Corollary (Roig, V., Weil, 2007)

Given a word $u \in F_k$, one can check whether u is primitive in F_k in time $O(n^2 k^3)$, where $n = \|u\|$.

Primitivity

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Observation

u is primitive $\Leftrightarrow$ the orbit of u contains a.

Corollary (Roig, V., Weil, 2007)

Given a word $u \in F_k$, one can check whether u is primitive in F_k in time $O(n^2 k^3)$, where $n = \|u\|$.

Theorem (Roig, V., Weil, 2007)

There is an algorithm which solves Whitehead Minimization Problem for F_k in time $O(n^2 k^3)$.

Observation

u is primitive $\Leftrightarrow$ the orbit of u contains a.

Corollary (Roig, V., Weil, 2007)

Given a word $u \in F_k$, one can check whether u is primitive in F_k in time $O(n^2 k^3)$, where $n = \|u\|$.

Deciding free-factorness

Observation

A given subgroup $H \leqslant F_k$ *of rank* $r(H) = r \leqslant k$ *is a free factor of* F_k *if and only if* $\exists\, \varphi \in Aut(F_k)$ *such that* $\|\varphi(H)\| = 1$.

Corollary (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$, *one can check whether* H *is a free factor of* F_k *in time* $O((n^2k^4 + n^3k^2)\log(nk))$, *where* $n = \|H\|$.

Corollary (Roig, V., Weil, 2007)

Given f.g. subgroups $H \leqslant K \leqslant F_k$, *one can check whether* H *is a free factor of* K *in polynomial time w.r.t. the given generators of* H *and* K.

Deciding free-factorness

Observation

A given subgroup $H \leqslant F_k$ of rank $r(H) = r \leqslant k$ is a free factor of F_k if and only if $\exists\, \varphi \in Aut(F_k)$ such that $\|\varphi(H)\| = 1$.

Corollary (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$, one can check whether H is a free factor of F_k in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Corollary (Roig, V., Weil, 2007)

Given f.g. subgroups $H \leqslant K \leqslant F_k$, one can check whether H is a free factor of K in polynomial time w.r.t. the given generators of H and K.

Deciding free-factorness

Observation

A given subgroup $H \leqslant F_k$ of rank $r(H) = r \leqslant k$ is a free factor of F_k if and only if $\exists\, \varphi \in Aut(F_k)$ such that $\|\varphi(H)\| = 1$.

Corollary (Roig, V., Weil, 2007)

Given a f.g. subgroup $H \leqslant F_k$, one can check whether H is a free factor of F_k in time $O((n^2k^4 + n^3k^2)\log(nk))$, where $n = \|H\|$.

Corollary (Roig, V., Weil, 2007)

Given f.g. subgroups $H \leqslant K \leqslant F_k$, one can check whether H is a free factor of K in polynomial time w.r.t. the given generators of H and K.

THANKS