Word problems and compression [CRM 2012/09/25]

Overview: We will discuss the word problem in $\mathrm{Aut}(\mathbb{F}_n)$ from an elementary and algorithmic point of view. Towards the end we'll generalize these ideas to $\mathrm{Aut}(G)$ where G is a Gromov hyperbolic group. Roughly the four lectures will be divided as follows.

I. Words, groups, Dehn's problems.

II. $\mathrm{Aut}(\mathbb{F}_n)$ via Nielsen reduction, Whitehead graphs, Stallings folds.

III. Compressed words, the algorithms of Plandowski, Hagenah, Lohrey.

IV. Gromov hyperbolic groups.

Please do ask questions; I'd rather be understood than finish all the material!

Lecture I

A) Words: Fix a finite alphabet $\mathcal{A} = \{a_1, a_2, \dots a_m\}$.

Fix $n \in \mathbb{N} = \{0, 1, 2, \dots\}$. A word of length n is a function $w : \{0, 1, \dots, n-1\} \longrightarrow \mathcal{A}$.

We write $|w| = n$ for the length of w.

Picture :

w_0	w_1	w_2	w_3	---	w_{n-2}	w_{n-1}		characters
$\uparrow$	$\uparrow$	$\uparrow$			$\uparrow$		$\uparrow$	
0	1	2		---	$n-1$		n	positions

Define the subword $u = w[i:j]$ by

$$u_k = w_{i+k}$$

for $k < j-i$. Thus $|u| =$ length of u.
$= j-i$

Note that our words are all zero-indexed.

We write $w[0:k] = w[:k]$ for the prefix of length k
and $w[k:n] = w[k:]$ for the suffix of length $n-k$.

We use ε to denote the (unique!) word of length zero: the empty word.

Define $a^* = \{ w \text{ a word over } a \}$ to be the Kleene closure of a: the set of all words.

Define also $u = \text{reverse}(w)$ by $u_i = w_{n-1-i}$

We will consider a^* as a monoid with concatenation serving as the multiplication: that is, if $u, v \in a^*$ then define $w = u \cdot v$ as follows.

If $w = u \cdot v$ then $w_i = \begin{cases} u_i & \text{if } i < |u| \\ v_{i-|u|} & \text{if } i \geq |u| \end{cases}$

Powers: For any word $w \in \mathcal{a}^*$ define w^n recursively, namely $w^0 = \varepsilon$ and $w^n = w^{n-1} \cdot w$, for $n \geq 1$.

Here is a more subtle example.

The Fibonacci words: Suppose $\mathcal{a} = \{a, b\}$. We take $f_1 = b$ and $f_2 = a$ [words of length one] and $f_k = f_{k-1} \cdot f_{k-2}$, for $k \geq 3$.

$f_1 = b$	$f_5 = abaab$
$f_2 = a$	$f_6 = abaababa$
$f_3 = ab$	$f_7 = abaababaabaab$
$f_4 = aba$	$f_8 =$ — — — —

We pause here to make our first remark about compression. Suppose we write n in binary; this takes appoximately $\log_2(n)$ bits. To write w takes $|w|$ characters, by definition. Thus we may regard the expression "w^n" to be a description, of total length $|w| + \log_2(n)$, of a word of total length $n \cdot |w|$. ~~Even more~~

Thus the expression ω^n is an "exponential compression."

<u>Exercise</u>: The Fibonacci words f_n offer doubly exponential compression.

<u>Exercise</u>: How many times does f_{10} appear as a subword of f_{25}? of f_{250}?

Suppose that $\mathcal{A}$ is an alphabet. Fix an ordering of the characters. Suppose u, v are words over $\mathcal{A}$ and p is the maximal common prefix. We have $u = p \cdot s$ and $v = p \cdot t$ for some words s, t. We say u is <u>short-lex before</u> v, and we write $u < v$ if either

(i) $|u| < |v|$ or

(ii) $|u| = |v|$ <u>and</u> $s_0 < t_0$.

(B) <u>Groups</u>: Here is our viewpoint: group elements are equivalence classes of words, where the equivalence relation is generated by insertion or deletion of <u>relators</u>. Here are the details.

Fix another alphabet $\bar{\mathcal{A}} = \{\bar{a}_1, \bar{a}_2, \cdots \bar{a}_m\}$ and let $\mathcal{S} = \mathcal{A} \cup \bar{\mathcal{A}}$. Let $\bar{\cdot} : \mathcal{S}^* \circlearrowleft$ be the involution given by *inverting*: $(\bar{\omega})_i = \overline{\text{reverse}(\omega)_i}$.

Let $T = \{ a\bar{a} \mid a \in \mathcal{S} \}$ be the set of *trivial* relations. If $u = xty$ and $v = xy$ for $t \in T$ and $x, y \in \mathcal{S}^*$, we write $u \xrightarrow{T} v$ and say u and v are related by a *free insertion* or *deletion*. Call $w \in \mathcal{S}^*$ *reduced* if w admits no free deletions. For any word $u \in \mathcal{S}^*$ let $[u]_T$ be the symmetric and transitive closure of the relation $\xrightarrow{T}$.

Define $\mathbb{F}_a$ to be the *free group* on $\mathcal{A}$: elements are $\{ [u]_T \mid u \in \mathcal{S}^* \}$, $[\varepsilon]_T$ is the identity element, $\bar{\cdot}$ gives inverses, and associtativity is an immediate consequence of associativity in the free monoid. There is still something to prove, however:

Theorem: Every class $[u]_T$ contains exactly one reduced word. *Proof*: Exercise.

In general, suppose $R \subset \mathcal{A}^*$ is a set of words. We write $u \xrightarrow{R} v$ if there are words $x, y \in \mathcal{A}^*$ and a <u>relation</u> $r \in R \cup T$ s.t. $u = xry$ and $v = xy$. As before take the symmetric and transitive closure of the relation $\xrightarrow{R}$ and let $[\cdot]_R$ be the resulting equivalence classes. We write $\langle a \mid R \rangle$ for the corresponding group.

Got Here sort of

The material above shows that $\langle a \mid R \rangle$ is obviously a group. What are <u>non</u>-obvious are the fundamental questions, as follows

---

Ⓒ <u>Dehn</u>: Fix a group $G = \langle a \mid R \rangle$. Recall that $\mathcal{A} = (a \cup \bar{a})$.

<u>Word Problem for G</u> <u>Instance</u>: $u, v \in \mathcal{A}^*$.

<u>Question</u>: $[u]_R = [v]_R$?

<u>Conjugacy Problem for G</u>: <u>Instance</u>: $u, v \in \mathcal{A}^*$.

<u>Question</u>: is there $w \in \mathcal{A}^*$ with $[wu\bar{w}] = [v]$?

Of somewhat different nature is

<u>Isomorphism problem</u>: <u>Instances</u> $G = \langle a \mid R \rangle, H = \langle B \mid S \rangle$

<u>Question</u>: Is G isomorphic to H?

(D) Normal forms: To answer such problems we attempt to find normal forms ~~that~~ are "efficiently computable". [Ideally using short-lex!]

A map $NF : \mathcal{S}^* \to \mathcal{S}^*$ is a normal form for $\langle a \mid R \rangle$ if

(i) $\forall w \in \mathcal{S}^*$, $NF(w) \in [w]_R$ (Existence)

(ii) $\forall u, v \in [w]_R$, $NF(u) = NF(v)$. (Uniqueness).

By the theorem, reduced words give a normal form for the free group. Exercise: Give an algorithm to find reduced forms of words. Code it up in your favorite computer language. [Try to beat quadratic time!]

Baumslag-Solitar Groups

$$BS(p,q) = \langle a, b \mid ba^p = a^q b \rangle.$$

[This is shorthand for $\langle \{a,b\} \mid \{ba^p\bar{b}\bar{a}^q\} \rangle$.]

We will check that, for $BS(1,2)$, the set

$$\left\{ \bar{b}^p a^q b^r \;\middle|\; \begin{array}{l} p,q,r \in \mathbb{Z} \text{ with } p,r \geq 0 \text{ and} \\ \text{if } p,r \geq 1 \text{ then } q \text{ is odd} \end{array} \right\}$$

is a collection of normal forms.

Here are some useful consequenses of the relator.

$$a\bar{b} \xrightarrow{R} \bar{b}a^2, \qquad ba \longrightarrow a^2 b$$

$$\bar{a}\bar{b} \longrightarrow \bar{b}\bar{a}^2, \qquad b\bar{a} \longrightarrow \bar{a}^2 b$$

We now analyze how the normal form must change when multiplied by a generator.

$p = r = 0$

$$a^q \cdot a^{\pm 1} \longrightarrow a^{q\pm 1}$$
$$a^q \cdot b \longrightarrow a^q b$$
$$a^q \cdot \bar{b} \longrightarrow \bar{b} a^{2q}$$

$p \geq 1, \ r = 0$

$$\bar{b}^p a^q \cdot a^{\pm 1} \longrightarrow \bar{b}^p a^{q\pm 1}$$
$$\bar{b}^p a^q \cdot b \longrightarrow \bar{b}^p a^q b \quad \text{if } q \text{ odd}$$
$$\longrightarrow \bar{b}^{p-1} a^{q/2} \quad \text{if } q \text{ even}$$
$$\bar{b}^p a^q \cdot \bar{b} \longrightarrow \bar{b}^{p+1} a^{2q}$$

$p = 0, \ r \geq 1$

$$a^q b^r \cdot a^{\pm 1} \longrightarrow a^{q\pm 2^r} b^r$$
$$a^q b^r \cdot b^{\pm 1} \longrightarrow a^q b^{r\pm 1}$$

$p, r \geq 1$ and q odd.

$$\bar{b}^p a^q b^r \cdot a^{\pm 1} \longrightarrow \bar{b}^q a^{p\pm 2^r} b^r$$
$$\bar{b}^p a^q b^r \cdot b^{\pm 1} \longrightarrow \bar{b}^p a^q b^{r\pm 1}$$

This verifies (i). We leave (ii) as an exercise.

[It suffices to check $NF(NF(\omega)\cdot r) = NF(\omega)$.]

for $r \in R \cup T$.

[Also: BS(1,2) is linear
Could use residually cyclic-by-cyclic.]

[Discuss complexity, models of computation, encodings]