Lecture III [CRM 2012/09/27]

A) WP(Aut(G)).

Just to review: If WP(G) is decidable then so is WP(H) for finitely generated $H < \mathrm{Aut}(G)$. This is because $\varphi \underset{H}{\equiv} \mathrm{Id}_H$ iff $\forall a \in \mathcal{A}$ we have $\varphi(a) \underset{G}{=} a$ and we can verify the latter. However

we typically pay an exponential price for this reduction: Consider $\varphi = i_a \circ i_b \circ \lambda_{ba} \circ i_a \circ \lambda_{ab}$

So:
$$a \to a \to \bar{a} \to \bar{a}\bar{b} \to \bar{a}b \to ab$$
$$b \to ab \to \bar{a}b \to \bar{a} \to \bar{a} \to a$$

and φ is the Fibonacci automorphism. Note $\varphi^n \circ \varphi^{-n}$ has length $10n$. To naïvely show $\varphi^n \circ \varphi^{-n} = \mathrm{Id}$ [as above] requires computing words in $\mathbb{F}_n$ of exponential length.

B) Straight line programs.

"A context free grammar that only produces one word."

Let $\mathbb{A} = \langle V, \mathcal{A}, A, P \rangle$ be a four-tuple with

- $A \in V$ the root [or axiom]
- $V = \{A_i\}_{i=1}^{n}$ the variables [non-terminal alphabet]

[typically $A = A_n$].

- $\mathcal{A} = \{a_j\}_{j=1}^{m}$ the terminal alphabet
- $\mathcal{P} = \{A_i \longrightarrow w_i\}_{i=1}^{n}$ the production rules

where we require $w_i \in (\{A_j\}_{j<i} \cup \mathcal{A})^*$.

The program $\mathbb{A}$ is in Chomsky normal form if $\forall i$ we have $|w_i| = 1$ or 2

- if $|w_i| = 1$ then $w_i \in \mathcal{A}$ is a terminal letter
- if $|w_i| = 2$ then $w_i \in V^*$

We alway use CNF: It makes the proofs much nicer.

Define $\text{eval}(\mathbb{A}) = w(\mathbb{A}) = w_{\mathbb{A}}$ to be the output of $\mathbb{A}$: the unique word produced by the grammar. Lets look at two examples.

val($\mathbb{A}$) $= w(\mathbb{A})$

Example: LaTeX macros \newcommand

Example: Squaring: we take

$$\mathbb{A} = \left\langle \{A_i\}_{i=1}^{n}, \{a\}, A_n, \{A_1 \to a, A_{i+1} \to A_i A_i\}_{i=2}^{n} \right\rangle.$$

We now compute: $A_n \to A_{n-1} A_{n-1} \to A_{n-2} A_{n-2} A_{n-2} A_{n-2} \to$

$$\cdots \longrightarrow \underbrace{aaaa \cdots a}_{2^{n-1} \text{ times}} = w_{\mathbb{A}}.$$

Define, for a fixed SLP $\mathbb{A}$, the <u>height</u> of A_i to be

$$\|A_i\| = \begin{cases} 0 & \text{if } A_i \to a \text{ for some } a \in \mathcal{A} \\ \max\{\|A_j\|, \|A_k\|\}+1 & \text{if } A_i \to A_j A_k \end{cases}$$

<u>Lemma</u>: For any $A_i \in V$ we have $|\mathrm{eval}(A_i)| \leq 2^{\|A_i\|}$.

<u>Pf</u>: Induct upwards. //

<u>Example</u>: $\mathbb{F}\mathrm{ib}_n = \left\langle \{F_i\}_{i=0}^{n}, \{a,b\}, F_n, \{F_0 \to b, F_1 \to a, F_k \to F_{k-1}F_{k-2}\}_{k=2}^{n} \right\rangle$

we compute

$$F_5 \longrightarrow F_4F_3 \longrightarrow F_3F_2F_3 \longrightarrow F_2F_1F_2F_2F_1 \longrightarrow$$
$$\longrightarrow F_1F_0F_1F_1F_0F_1F_0F_1 \longrightarrow abaababa = w(F_5)$$
$$= \varphi^5(b).$$

<u>Thus</u>, we do <u>not</u> want to run SLP's!

<u>Some exercises</u>: Give polynomial-time algorithms that

- Given $\mathbb{A}$, compute $|w_B|$ for all $B \in V$.
- Given $\mathbb{A}$, $i \in \mathbb{N}$ compute $w_A[i]$ (the ith character)
- Given $\mathbb{A}$, $a \in \mathcal{A}$ count the number of times 'a' appears in w_A.

for SLP-polynomials (use $+, \times, \mathbb{1}, x$) can ① evaluate at $p \in \mathbb{Z}$ ② add, multiply such ③ Differentiat

Here is a deeper result:

Plandowski's Algorithm [~~[illegible]~~]

There is a polynomial-time algorithm that, given SLP's $\mathbb{A}, \mathbb{X}$ over $\mathcal{A}$, computes the maximal k (in binary) so that $w_{\mathbb{A}}[ik] = w_{\mathbb{X}}[ik]$.

If there is time we may discuss this at the end of the talk. Just notice that the exercises and Plan. Alg. require you to learn things about the output $w_{\mathbb{A}}$ without computing $w_{\mathbb{A}}$.

(c) Compressed group elements: Suppose $G = \langle \mathcal{A} \mid R \rangle$ is our group. Let $\mathbb{A}$ be an SLP over the terminal alphabet $\mathcal{A} = \mathcal{A} \cup \bar{\mathcal{A}}$. We call $\mathbb{A}$ a compressed word

The compressed word problem over G is:

CWP(G): Instance: $\mathbb{A}, \mathbb{X}$ compressed words

Question: Is $w_{\mathbb{A}} \underset{G}{=} w_{\mathbb{X}}$?

Exercise
~~[illegible]~~
Given $\mathbb{A}$ find $\bar{\mathbb{A}}$.

Note this is decidable iff WP(G) is decidable. One way to solve this is to rewrite $\mathbb{A}$ so it instead outputs a normal form.

Thm [Lohrey] The CWP(F_m) has a poly-time solution.

Thm [Lohrey] There is a poly-time algorithm that, given $\mathbb{A}$ over $\mathcal{A}$ computes an SLP $\mathbb{X}$ so that

- $w_{\mathbb{A}} =_G w_{\mathbb{X}}$ and
- $w_{\mathbb{X}}$ is reduced.

(D) Composition systems: These are a technically neat version of SLP's: they make Lohrey's proof work.

If $A, B, C \in V$ then we allow production rules of the form $A \to B[i:j] \cdot C[k:l]$. Here $B[i:j]$ is a truncated non-terminal; these only appear on right hand sides of rules, never on the left. We require $0 \le i \le j \le |w_B|$. Since the intent is $w(B[i:j]) = w_B[i:j]$ we define repeated truncation: $(B[i:j])[k:l] = B[i+k : i+l]$ [assuming $l \le j-i$].

However, composition systems are not really more powerful than SLP's.

<u>Thm</u> [Hagenah] There is a polynomial time algorithm that, Given a comp. system $\mathbb{A}$, returns an SLP $\mathbb{X}$ s.t. $\omega_{\mathbb{A}} = \omega_{\mathbb{X}}$. [Also the SLP $\mathbb{X}$ is only quadratically larger.]

<u>Corollary</u>: Plandowski's Algorithm applies to composition systems.

(E) <u>Proof sketch of Lohrey's Algorithm.</u> We are given an SLP $\mathbb{A}$ over $\mathcal{F}$. We must produce a SLP $\mathbb{X}$ st. • $\omega_{\mathbb{A}} =_F \omega_{\mathbb{X}}$ and • $\omega_{\mathbb{X}}$ is reduced.

We will instead produce a composition system with those properties; we work bottom up.

Write $\mathbb{A} = \langle V_A, \mathcal{F}, A, P \rangle$ and $\mathbb{X} = \langle V_X, \mathcal{F}, X, Q \rangle$.

For each $A_i \in V_A$ of height one, add a copy X_i to V_X and if $A_i \to a_j$ then add $X_i \to a_j$ to Q.

<u>Induction step</u>: Suppose $A \to B \cdot C$ lies in P.

Then we are given $Y, Z \in V_X$ st.

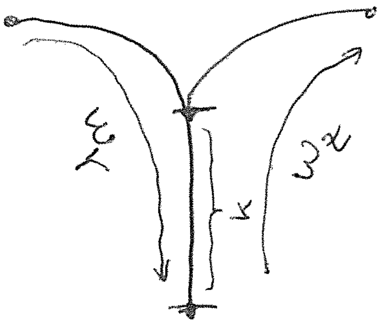
- $\omega_B = \omega_Y$, ω_Y reduced
- $\omega_C = \omega_Z$, ω_Z reduced.

Thus $\omega_A =_F \omega_B \cdot \omega_C =_F \omega_Y \cdot \omega_Z$

So add X to V_X. We now compute the production rule for X. Build the composition system with root $\bar{Y}$. Use Plandowski's algorithm to compute $k \in \mathbb{N}$, the length of the maximal common prefix for $w(\bar{Y})$ and $w(Z)$.

Thus the word $w_Y[:-k] \cdot w_Z[k:]$ is reduced.

Picture:



So add $X \longrightarrow Y[:-k] \cdot Z[k:]$ to Q and quit. //

(F) WP(Aut(F_m)): The word problem for Aut(F_m) is poly-time. Write $F_m = F(a_1, a_2, \dots, a_m)$.

Proof: Write $\varphi = \varphi_1 \circ \varphi_2 \circ \varphi_3 \circ \dots \circ \varphi_n$ a composition of Nielsen transformations. We now construct a straight line program.

Let $V_A = \{ A_{i,p}, \bar{A}_{i,p} \}$ with $i \in \{1, 2, \dots, m\}$, $p \in \{0, 1, 2, \dots, n\}$.

We use the following production rules:

$A_{i,0} \longrightarrow a_i$, $\bar{A}_{i,0} \longrightarrow \bar{a}_i$ and

$$A_{i,p} \longrightarrow W_{i,p}, \quad \overline{A}_{i,p} \longrightarrow \overline{W}_{i,p}$$

where $W_{i,p} \in \{ A_{i,p-1}, \overline{A}_{i,p-1} \mid i \in \{1, 2, \cdots m\} \}^*$

is the word corresponding to the image of the Nielsen transform φ_p applied to a_i.

[capitalize and add p-1 as subscript.]

Now apply Lohrey's algorithm to $\mathbb{A}$ to make all outputs reduced. Check that $\mathrm{eval}(A_{i,n}) = a_i$

If this holds for all i, then $\varphi = \mathrm{Id}_F$, as desired.

//

Questions: ① Write up the compressed conj problem for A_Γ (RAAG)?

② Normal forms for $\varphi \in \mathrm{Aut}(\mathbb{F}_n)$? [As usual, the conj problem for $\mathrm{Aut}(\mathbb{F}_n)$]

③ Think more about the fully compressed membership problem? [See paper of Artur Jeż "compressed membership"]

④ Benoit (sp) has a version of Stallings folds (in CS - reducing automata??)
[Benois (sp?)]

⑤ [Lohrey: CWP for the braid group]