1 The Proof Certificate

An element of a group G is equal to one in the quotient by the normal closure of a relation r if and only if it can be written as a product of conjugates of r and r^{-1}. More precisely, there is a group homomorphism $e : F(G) \to G$, from the free group over G into G that sends a basis element of $F(G)$, $g \in G$ to $grg^{-1} \in G$. The image of this map is exactly the kernel of the quotient map. Therefore an element p of $F(G)$ such that $e(p) = w$ can be seen as a witness that w is in the kernel of the quotient map.

The algorithm returns a pair of a normalised word w', and $p \in F(G)$ such that $w = e(p)w'$.

Definition 1.1 (P functor)**.** We define a group structure on this set of pairs. For any $g \in G$ define an automorphism $MulFree(g)$ of $F(G)$, by sending a basis element $h \in G$ to gh. This defines a left action of G on $F(G)$. The group structure is given by the semidirect product. Define the group $P(G)$ to be

$$P(G) := F(G) \rtimes_{MulFree} G \tag{1}$$

This group has multiplication given by $(a, b)(a', b') = (aMulFree(b)(a'), bb')$

Definition 1.1.1 (lhs and rhs)**.** We define two group homomorphisms from P into G. rhs is the obvious map sending (a, b) to b. lhs is the map sending (a, b) to $e(a)b$. Since $e(a)$ is in the kernel of the quotient map, for any $p \in P(G)$, $\mathrm{lhs}(p)$ and $\mathrm{rhs}(p)$ are equal in the quotient by r. Therefore an element p of $P(G)$ can be regarded as a certificate of the congruence $\mathrm{lhs}(p) = \mathrm{rhs}(p) \bmod r$.

Because both lhs and rhs are group homomorphisms, if $p \in P(G)$ is a certificate of the congruence $a = b \bmod r$, and q is a certificate of the congruence $c = d \bmod r$, then pq is a certificate of the congruence $ac = bd \bmod r$. Similarly p^{-1} is a certificate of the congruence $a^{-1} = b^{-1} \bmod r$.

Definition 1.1.2. (P is functorial). Given a homomorphism $f : G \to H$. Functoriality of the free group gives a natural map $F(f) : F(G) \to F(H)$ from $f : G \to H$. Define the map $P(f) : P(G) \to P(H)$ to send $(p, b) \in P(G)$ to $(F(f)(p), f(b)) \in P(H)$. Given a certificate of the congruence $a = b \bmod r$, this map returns a certificate of the congruence $f(a) = f(b) \bmod f(r)$.

Definition 1.1.3. (Trans) Given $p, q \in P(G)$ such that p is a certificate of the congruence $a = b \bmod r$, and q is a certificate of the congruence $b = c \bmod r$, then it is possible to define $\mathrm{Trans}(p, q)$ such that $\mathrm{Trans}(p, q)$ is a certificate of the congruence $a = c \bmod r$. If $p = (p_1, p_2)$, and $q = (q_1, q_2)$, then $\mathrm{Trans}(p, q) = (p_1q_1, q_2)$.

Definition 1.1.4. (Refl) Given $a \in G$, $(1, a)$ is a certificate of the congruence $a = a \bmod r$. Call this $\mathrm{Refl}(a)$.

It is also possible to define *Symm*, such that $\mathrm{lhs}(\mathrm{Symm}(p)) = \mathrm{rhs}(p)$ and vice versa, but this is not used in the algorithm.

Definition 1.1.5. (ChangeRel) Given a certificate p of the congruence $a = b \bmod r$, then it is possible to make a certificate of the congruence $a = b \bmod grg^{-1}$ for any $g \in G$. This

2 Adding and Removing Subscripts

Given a letter t in the free group over a set S, we can define a map into a semidirect product.

Definition 2.1 (ChangeSubscript)**.** Define a homomorphism *ChangeSubscript* from $\mathbb{Z}$ to the automorphism group of $F(S \times \mathbb{Z})$. If $(x, n) \in S \times mathbbZ$ is a basis element of the free group, then $\text{AddBasis}(m)(x, n) = (x, m + n)$.

Definition 2.2 (AddSubscripts)**.** There is a homomorphism *AddSubscripts* from $F(S)$ into $F(S \times \mathbb{Z}) \rtimes_{\text{ChangeSubscript}} \mathbb{Z}$ sending a basis element $s \in S$ to $(s, 0) \in F(S \times \mathbb{Z})$, when $s \neq t$ and sending t to $(1, 1_{\mathbb{Z}}) \in F(S \times \mathbb{Z}) \rtimes \mathbb{Z}$. Loosely, this map replace occurence of $t^n a t^{-n}$ with a_t

The map *AddSubscripts* is only used during the algorithm on words w when the sum of the exponents of t in w is zero, meaning the result will be of the form $(w', 0_{\mathbb{Z}})$.

Definition 2.3 (RemoveSubscripts)**.** *RemoveSubscripts* send a basis element of $F(S \times \mathbb{Z})$, $(s, n) \in S \times \mathbb{Z}$ to $t^n s t^{-n}$.

RemoveSubscripts is a group homomorphism and if r is a word such that of $\text{Addsubscripts}(r)$ is of the form $(r', 0_{\mathbb{Z}})$, then $\text{RemoveSubscripts}(r') = r$.

3 HNN Extensions

Definition 3.1 (HNN Extension)**.** Given a group G and subgroups A and B of G, and an isomorphism $\phi : A \to B$, we can define the *HNN extension* of G. Let $\langle t \rangle$ be a multiplicative group isomorphic to $\mathbb{Z}$, generated by t. The HNN extension is the coproduct of G and $\langle t \rangle$ quotiented by the normal closure of the set $\{tat^{-1} = \phi(a) | a \in A\}$

Theorem 3.2 (Britton's Lemma)**.** Let $w = g_0 t^{k_1} g_1 t^{k_2} g_2 \cdots t^{k_n} g_n$ be a word in the HNN extension. If for every i, $k_i \neq 0$, $k_i > 0 \implies g_i \notin A$ and $k_i < 0 \implies g_i \notin B$, and w contains a t, then $w \neq 1$ cite (On Britton's theorem Charles Miller)

Corollary 3.2.1. If a word w meets the same conditions as in the statement of Britton's Lemma, then w cannot be written as a t free word.

Proof. Suppose $w = g$ with $g \in G$, then $g^{-1}w$ also meets the conditions in Theorem 3.2, and therefore $g^{-1}w \neq 1$, contradicting $w = g$.

The HNN normalization process replaces any occurences of ta with $\phi(a)t$ when $a \in A$, and any occurance of $t^{-1}b$ with $\phi^{-1}(b)t^{-1}$ when $b \in B$. This produces a word of the form in Theorem 3.2.

4 Base Case

The base case is the case where the relation r is of the form a^n with $n \in \mathbb{Z}$, and a a letter in S. It is straightforward to decide the word problem in this group, since $F(S)/a^n$ is isomorphic to the binary coproduct of $F(S \backslash \{a\})$ and $\mathbb{Z}/n\mathbb{Z}$. However computing the appropriate proof term requires some explanation.

5 HNN normalization

We first present a simplified version of the HNN normalization that does not compute the proof certificates, and then explain how to compute the certficates at the same time as normalization.

To compute the HNN normalized term, first compute the following isomorphism from $F(S)$ into the binary coproduct $F(S'') * \langle t' \rangle$. $\langle t' \rangle$ is a cyclic multiplicative group isomorphic to $\mathbb{Z}$ and generated by t'.

Definition 5.1. Define a map on a basis element i as follows

$$\begin{cases} (i, 0) \in F(S'') & i \neq t \\ t' & i = t \end{cases} \tag{2}$$

It is important that $a \leq 0 \leq b$, to ensure that this map does map into $F(S'' \times \langle t' \rangle)$.

Then apply the HNN normalization procedure. For this particular HNN extension ϕ is *ChangeSubscript* (Definition 2.1). For each occurence of gt'^{-1}, we can use *Solve* to check whether g is equal to a word $a \in A$ in the quotient $F(S'')/r'$, and if it is equal to some word a, rewrite gt'^{-1} to t'^{-1}Changesubscript$(-1)(a)$. Similarly, For each occurence of gt', use *Solve* to check whether g is equal to a word $b \in B$ in the quotient $F(S'')/r'$, and if it it equal to some word b rewrite gt' to t'ChangeSubscript$(1)(b)$.

This is a slight deviation from the procedure decscribed in Section 3, the rewriting rules are applied in the opposite direction, and therefore the words are not quite in the normal form described in Theorem 3.2, but are in a reversed version of this normal form. The rewriting rules are applied from left to right; wherever a rewriting rule is applied, everything to the left of where that rule is applied should already be in HNN normal form. The reasons for these are to generate shorter certificates and are described in Section 5.1.

5.1 Computing Proof Certificates

To compute proof certificates a slightly modification of the procedure described in Section 5 is used.

First define a modification of Definition 5.1, from $F(S)$ into the binary coproduct $P(F(S \times \mathbb{Z})) * \langle t' \rangle$.

Definition 5.2. Define a map on the basis as follows

$$\begin{cases} \text{Refl}(i, t'^0) \in F(S'' \times \langle t' \rangle) & i \in S \text{ and } i \neq t \\ t' & i = t \end{cases} \tag{3}$$

There is also a map Z from $P(F(S \times \mathbb{Z})) * \langle t' \rangle$ into $P(F(S))$. This map is not computed as part of the algorithm, but is useful to define anyway.

Definition 5.3. The map Z sends $t' \in \langle t' \rangle$ to $\text{Refl}(t) \in P(F(S))$. It sends $p \in P(F(S \times \mathbb{Z}))$ to $P(\text{RemoveSubscripts})(p) \in P(F(S))$

The aim is to define a normalization process into that turns a word $w \in F(S)$ into word $n \in P(F(S \times \mathbb{Z})) * \langle t' \rangle$, such that after applying *rhs*, the same word is returned as in the normalization process described in Section 5. We also want $\text{lhs}(Z(n))$ to be equal to w, so we send up with a certificate that w is equal to some normalized word.

Definition 5.4. (conjP) Let $(p, a) \in P(F(S \times \mathbb{Z}))$ and $k \in \mathbb{Z}$. Define *ConjP* to map into $P(F(S \times \mathbb{Z}))$

$$\text{ConjP}(k, (p, a)) = (\text{MulFree}((t, 0)^k, p), \text{ChangeSubscript}(k, a)) \tag{4}$$

conjP has the property that $\text{lhs}(Z(\text{conjP}(k, p))) = t^k \text{lhs}(Z(p)) t^{-k}$, and similarly for *rhs*. Note that *conjP* maps into $P(F(S \times \mathbb{Z}))$, and not $P(F(S''))$, although *rhs* of every word computed will be in $F(S'')$.

The procedure described in Section 5 replaced each occurence of wt'^{-1} with $t'^{-1}\text{ChangeSubscript}(-1)(a)$, where $a \in A$ was a word equal to $w \in F(S'')$ in the quotient $F(S'')/r'$.

To compute the proof certificate suppose there is an occurence of pt'^{-1} with $p \in P(F(S \times \mathbb{Z}))$. Use *Solve* to check whether $\text{rhs}(p)$ is equal to a word $a \in A$. Suppose $q \in P(F(S''))$ is the certificate of this congruence. Then substitute pt'^{-1} with $t'^{-1}\text{ConjP}(1, \text{Trans}(p, q))$.

Similarly replace for every occurence of pt', with $\text{rhs}(p)$ equal to some word $b \in B$, and q a certificate of this congruence, replace pt' with $t'\text{ConjP}(-1, \text{Trans}(p, q))$.

5.1.1 Proof Lengths

It is important to apply the rewriting procedure from left to right. This will usually produce shorter proof certificates. Consider normalizing $t'^{-1}wt'vt'^{-1}$ with $w, v \in F(S'')$. Suppose we normalize right to left. $t'v$ is normalized to pt' with $p \in P(F(S \times \mathbb{Z}))$. After that substitution the new word $t'^{-1}wp$, with $wp \in P(F(S \times \mathbb{Z}))$. Suppose $\text{rhs}(wp)$ is normalized to $q \in P(F(S \times \mathbb{Z}))$. Then the final normalized word is $\text{conjP}(1, \text{Trans (wp, q)})t'^{-1}$.

The proof part of $\text{Trans}(wp, q)$ is $\text{MulFree}(w)(\text{Left}(p))\text{Left}(q)$.

Normalizing the other way, we first normalize $t'^{-1}w$ to $p_2t'^{-1}$ with $p_2 \in P(F(S \times \mathbb{Z}))$. Then the new word is p_2vt^{-1}. Suppose $\text{rhs}p_2v$ is normalized to $q_2 \in P(F(S \times \mathbb{Z}))$. Then the final normalized word is $t'^{-1}\text{conjP}(-1, \text{Trans}(p_2v, q_2))$.

The proof part of $\text{Trans}(p_2v, q)$ is $\text{Left}(p_2)\text{Left}(q)$. There is no use of *MulFree* since $\text{Left}(v) = 1$. On average using *MulFree* will make words longer, and so the left to right normalization produces shorter proofs. The left to right normalization was found to produce shorter proofs in practice.