Glasgow Math. J. **57** (2015) 173–180. © Glasgow Mathematical Journal Trust 2014.
doi:10.1017/S0017089514000202.

COMPUTING WITH SUBGROUPS OF THE MODULAR GROUP

MARKUS KIRSCHMER
*Lehrstuhl D für Mathematik, RWTH Aachen University,
Templergraben 64, 52062 Aachen, Germany
e-mail: Markus.Kirschmer@math.rwth-aachen.de*

and CHARLES LEEDHAM-GREEN
*School of Mathematical Sciences,
Queen Mary College University of London,
Mile End Road, London E1 4NS, United Kingdom
e-mail: C.R.Leedham-Green@qmul.ac.uk*

(Received 20 November 2012; revised 4 July 2013; accepted 3 December 2013; first published online 26 August 2014)

Abstract. We give several algorithms for finitely generated subgroups of the modular group $\mathrm{PSL}_2(\mathbb{Z})$ given by sets of generators. First, we present an algorithm to check whether a finitely generated subgroup H has finite index in the full modular group. Then we discuss how to parametrise the right cosets of H in $\mathrm{PSL}_2(\mathbb{Z})$, whether the index is finite or not. Further, we explain how an element in H can be written as a word in a given set of generators of H.

2010 *Mathematics Subject Classification.* 20-04, 20H05

1. Introduction. There exist several ways to describe a finitely generated subgroup H of the modular group $\mathrm{PSL}_2(\mathbb{Z})$. First of all, as a set of generating matrices or a set of generators given as words in some distinguished generators of $\mathrm{PSL}_2(\mathbb{Z})$. Another way is as follows. The group $\mathrm{PSL}_2(\mathbb{Z})$ acts on the right cosets of H in $\mathrm{PSL}_2(\mathbb{Z})$ by right multiplication. So if the index $k := [\mathrm{PSL}_2(\mathbb{Z}) : H]$ of H in $\mathrm{PSL}_2(\mathbb{Z})$ is finite, we obtain a permutation representation from $\mathrm{PSL}_2(\mathbb{Z})$ to the symmetric group on k letters.

In [**3**], Hsu gave a very efficient criterion to decide whether H is a congruence subgroup (i.e. it contains the kernel of the canonical epimorphism $\mathrm{PSL}_2(\mathbb{Z}) \to \mathrm{PSL}_2(\mathbb{Z}/N\mathbb{Z})$ for some integer $N \geq 2$), provided that H is given by a permutation representation. However, if H is given by a finite set of generators, it is not obvious how to decide even whether the index $[\mathrm{PSL}_2(\mathbb{Z}) : H]$ is finite or not.

In the present paper, algorithms are presented to solve these problems not only for $\mathrm{PSL}_2(\mathbb{Z})$ but in a slightly more general setting.

Let $G = \langle g_1, \dots, g_e \rangle$ be a finitely generated group. An element $g \in G$ is always assumed to be given as a word in the generators $g_1, \dots, g_e$ if not stated otherwise. Then $\|g\|$ denotes the length of this word.

Suppose further that the following conditions hold.

(1) G contains a non-abelian normal subgroup N of finite index that is free of finite rank, say r. Let $\{x_1, \dots, x_r\}$ be a basis of N.
(2) There exists a transversal $(t_1 = 1, t_2, \dots, t_f)$ of N in G (i.e. G is a disjoint union of cosets $t_iN = Nt_i$) and an Algorithm (A) that, given $g \in G$, returns some $A(g) \in \{t_1, \dots, t_f\}$ such that $gN = A(g)N$.

(3) There exists an Algorithm (B) that, given $g \in N$ (as a word in $g_1, \dots, g_e$), writes g as a word $B(g)$ in $\{x_1, \dots, x_r\}$.

THEOREM 1.1. *Let $H = \langle h_1, \dots, h_m \rangle$ be a finitely generated subgroup of G. Then there exist a transversal S_H for the right cosets of H in G and algorithms to solve the following problems.*

(1) *Decide if H has finite index in G, and if so, compute the index* $[G : H]$.
(2) *Given $g \in G$, return $s \in S_H$ such that $Hs = Hg$.*
(3) *Given $g \in H$, write g as a word in $\{h_1, \dots, h_m\}$.*

In particular, part 2 of the above theorem implies that membership in subgroups of G is decidable. Further, these algorithms have been implemented by the authors in Magma (see [**2**]) and can be obtained from the homepage of the first author.

Group elements are assumed to be represented by words in some generators. Let us assume that copying or deleting a single generator or comparing two generators takes constant time. In this model, forming the product of two words of length $\le n$ takes time $O(n)$.

THEOREM 1.2. *Suppose the notation of Theorem 1.1 and suppose further that Algorithms (A) and (B) satisfy the following conditions.*

- *Algorithms (A) and (B) run in polynomial time in $\|g\|$ when applied to some $g \in G$.*
- *There exists some constant $c \in \mathbb{N}$ such that, for all $g \in N$, the length of $B(g)$ as a word in $\{x_1, \dots, x_r\}$ is at most $c \cdot \|g\|$.*

Then the first two problems of Theorem 1.1 can be solved in polynomial time in c and the size of the input.

It is worthwhile to mention that a constant c as in the previous theorem need not exist. Even in the case where $G = N$ is free, writing elements given as words in one set of generators as words in another set of generators might have exponential growth as shown by Example 3.7.

The paper is organised as follows. In Section 2 the concept of Nielsen reduced sets is recalled. Section 3 contains the algorithms claimed in Theorem 1.1 as well as a proof of Theorem 1.2. Finally, Section 4 applies the above theorems to the special linear group $\mathrm{SL}_2(\mathbb{Z})$ and the modular group $\mathrm{PSL}_2(\mathbb{Z})$.

2. Free groups.

In this section we recall the concept of Nielsen generators for free groups.

Let F be a free group with basis $\{x_1, \dots, x_r\}$ for some $r > 1$. A word in these generators is said to be *reduced* if it does not contain a substring of the form $x_i x_i^{-1}$ or $x_i^{-1} x_i$. Every element $g \in F$ is represented by a unique reduced word $\rho(g)$ in the x_i, and we denote by $|g|$ the length of the word $\rho(g)$. We assume that each element in F is given as a reduced word in the x_i if not stated otherwise.

DEFINITION 1.

(1) If $V \subset F$, we write $V^{\pm}$ for the set $V \cup \{v^{-1} \mid v \in V\}$.
(2) A finite subset V of F is said to be *Nielsen reduced* if the following conditions hold.
   - $1 \notin V$,
   - $gh = 1$ or $|gh| > |g|$ for all $g, h \in V^{\pm}$,

- $gh = 1$ or $hi = 1$ or $|ghi| > |g| - |h| + |i|$ for all $g, h, i \in V^{\pm}$.

(3) Let V be a Nielsen reduced set. Suppose $v := gh \in V^{\pm}$ such that $2|g| = 2|h| = |v|$. Then h is said to be *isolated* if v is the only element in $V^{\pm}$ with terminal segment h.

(4) A Nielsen reduced set V is said to be *normalised* if the right halves of all elements of even length in V are isolated.

(5) If V is a normalised Nielsen reduced set, then $\mathcal{T}(V)$ denotes the set of all $g \in F$ such that
- g is an initial segment of some $v \in V$ such that $|g| \leq |v|/2$, or
- g^{-1} is a terminal segment of some $v \in V$ such that $|g| < |v|/2$.

Note that we do allow g to be the identity element of F. Thus, $\mathcal{T}(V)$ will never be empty.

Every finitely generated subgroup of F is generated by some normalised Nielsen reduced set V. To state an algorithm that computes such a set V, one needs to define a total order $<$ on F.

Given $g, h \in F$ as reduced words in the x_i, we define

$$g < h \iff |g| < |h| \text{ or } (|g| = |h| \text{ and } g <_l h),$$

where $<_l$ denotes the lexicographical order satisfying

$$x_1 <_l x_1^{-1} <_l x_2 <_l x_2^{-1} <_l \dots .$$

The minimum of two elements of F shall always refer to the minimum with respect to $<$.

ALGORITHM 2.1. ([**1**, Algorithm 1])*Given a finite set $W \subset F$, this algorithm computes a normalised Nielsen reduced set $V \subset F$ such that $\langle W \rangle = \langle V \rangle$.*

(1) *Replace each $w \in W$ by* $\min\{\rho(w), \rho(w^{-1})\}$. *Then remove all copies of the empty word from the set W.*

(2) *If there exist $v \neq w \in W$ and $\mu, \nu \in \{\pm 1\}$ such that $|v^{\mu} w^{\nu}| < |v|$, then: if $v^{\mu} w^{\nu}$ is the identity, remove v from W, otherwise replace v by* $\min\{\rho(v^{\mu} w^{\nu}), \rho(w^{-\nu} v^{-\mu})\}$ *and repeat this step.*

(3) *Take the least element $v = pq^{-1} \in W$ (with respect to $<$) such that:*
- $2|p| = 2|q| = |v|$.
- *There exists some $w \in W$ with $v \neq w$ that has q as initial or q^{-1} as terminal segment.*

If v exists, replace w by

$$\begin{cases} \min\{\rho(vw), \rho(w^{-1}v^{-1})\} & \text{if } q \text{ is the initial segment of } w, \\ \min\{\rho(vw^{-1}), \rho(wv^{-1})\} & \text{otherwise,} \end{cases}$$

and go to (2). If v does not exist, return $V := W$.

As explained in [**1**, pp 65–66], the algorithm gives correct output and runs in time $O(n^4 \cdot (\# W)^2)$, where n denotes the length of the largest word in W.

THEOREM 2.2 (Karrass and Solitar). *Let F be a free group of finite rank $r > 1$, and let V be a finite normalised Nielsen reduced subset of F. Further, let $\mathcal{T}(V)$ be as in*

Definition 1. Then $\langle V\rangle$ has finite index in F if and only if

$$\#\mathcal{T}(V)\cdot(r-1)=\#V-1\,.$$

Further, if the index is finite, it equals $\#\mathcal{T}(V)$.

Proof. See [**4**, Theorem 4]. □

It is clear that, given V, one can compute the set $\mathcal{T}(V)$ in time $O(n^3\cdot(\#V)^2)$, where n denotes the length of the largest word in V.

LEMMA 2.3. *Suppose $V\subset F$ is a normalised Nielsen reduced set. Then*

$$S(V):=\{g\in F\mid |vg|>|g|\ \textit{for all}\ v\in V\ \textit{and}\ |v^{-1}g|\geq|g|\ \textit{for all}\ v\in V\}$$

is a system of representatives for the right cosets of $\langle V\rangle$ in F.

Proof. See for example [**1**, Lemma 3.1]. □

ALGORITHM 2.4. *Given $g\in F$ and a normalised Nielsen reduced set $V\subset F$, this algorithm returns a word w in the elements of $V^{\pm}$, and $s\in S(V)$, such that $g=ws$.*

(1) *Initialize $(w,g)=(1,\rho(g))$.*
(2) *While there exists some $v\in V^{\pm}$ such that $|vg|<|g|$ or there exists some $v\in V$ such that $|vg|=|g|$, replace (w,g) by $(wv^{-1},\rho(vg))$.*
(3) *Return w and $s:=g$.*

It is clear that if the algorithm terminates, then the returned values s and w satisfy $g=ws$ and $s\in S(V)$. Suppose that g is given as a (not necessarily reduced) word in the x_i of length of at most n. Then the algorithm terminates after at most $n+1$ iterations as explained in [**1**, p. 69]. Since the computation of $\rho(g)$ runs in time $O(n)$ and each iteration in step 2 has cost $O(n\cdot\#V)$, the total cost is $O(n^2\cdot\#V)$.

3. Algorithms.

Assume the notation of Section 1. Then G acts on the cosets $N, t_2N,\dots,t_fN$ by left multiplication. Clearly, the stabiliser of N under the action of the subgroup $H\subset G$ equals $N\cap H$ and the union of the cosets in the H-orbit of N is HN.

As in the case of free groups, the algorithms claimed in Theorem 1.1 require some preprocessing stage, which we state first.

ALGORITHM 3.1 *Preprocessing.*

(1) *Using orbit enumeration and Algorithm (A), compute the following.*
    (a) *A set W of generators of $N\cap H$ as words in $\{h_1,\dots,h_m\}$.*
    (b) *For each $1\leq j\leq f$, store*

$$i_j=\min\{1\leq i\leq f\mid t_jN\ \textit{and}\ t_iN\ \textit{are in the same}\ H\textit{-orbit}\}$$

and some $\tilde{h}_j$ as a word in $\{h_1,\dots,h_f\}$ such that $t_jN=\tilde{h}_jt_{i_j}N$.

(2) *Using Algorithm (B), write the elements of $W\subset N$ as words in the x_i.*
(3) *Compute a normalised Nielsen reduced set V (as words in the x_i) generating $\langle W\rangle=N\cap H$ with Algorithm 2.1.*

REMARK 3.2. Suppose the situation of the previous algorithm and let $n = \sum_{i=1}^{m} \|h_i\| + \sum_{j=1}^{f} \|t_j\|$. Further, suppose that Algorithms (A) and (B) satisfy the assumptions of Theorem 1.2.

- The orbit enumeration in step 1 multiplies each t_j with each h_i. Since both elements have length of at most n, the costs of this step are $O(nmf)$. Further, this step makes fm calls to Algorithm (A), each time with words of length $\leq 2n$ as input. The elements $\tilde{h}_j$ are products of at most f elements from $\{h_i^{\pm 1} \mid 1 \leq i \leq m\}$.
- Let $k := \#\{1 \leq j \leq f \mid i_j = 1\}$. The second step calls Algorithm (B) at most $\#W \leq km \leq fm$ times with input words of length of at most fn. The words returned by Algorithm (B) thus have length $\leq fnc$ in $\{x_1, \ldots, x_r\}$.
- The third step runs in time $O((fnc)^4 (fm)^2)$ as explained in Section 2.

Thus, the preprocessing runs in polynomial time in c and the size of the input.

We are now ready to give the first algorithm claimed in Theorem 1.1.

ALGORITHM 3.3. *The following algorithm computes the index of H in G.*

(1) *From the preprocessing, get $k := \#\{1 \leq j \leq f \mid i_j = 1\}$ and V.*
(2) *From V, compute the set $\mathcal{T}(V)$ as in Definition 1.*
(3) *Using Theorem 2.2, decide whether $N \cap H = \langle V \rangle$ has finite index in N. If not, return ∞. Otherwise, return $\#\mathcal{T}(V) \cdot f/k$.*

Proof. The group H acts on G/N. The stabiliser of N is $N \cap H$. Thus, k equals the index of $N \cap H$ in H. Further,

$$[G : H] \cdot k = [G : H] \cdot [H : N \cap H] = [G : N] \cdot [N : N \cap H] = f \cdot [N : N \cap H] .$$

Now if $[N : N \cap H]$ is finite, it equals $\#\mathcal{T}(V)$ by Theorem 2.2. □

As an immediate consequence, one obtains the following corollary.

COROLLARY 1. *If $H = \langle h_1, \ldots, h_m \rangle$ is a finite index subgroup of G, then*

$$[G : H] < mf/(r-1) .$$

Proof. Assume the notation of Algorithm 3.3. Then $[G : H] = f/k \cdot \#\mathcal{T}(V)$ (*loc. cit.*), and Theorem 2.2 shows that $\#\mathcal{T}(V) = (\#V - 1)/(r-1)$. Further, $\#V \leq \#W \leq km$ by Remark 3.2. Thus,

$$[G : H] \leq f/k \cdot (km-1)/(r-1) < fm/(r-1) .$$

□

Now we turn to the other algorithms of Theorem 1.1. For this, we need to define a system of representatives of the right cosets of H in G similar to Lemma 2.3.

LEMMA 3.4. *Suppose the notation of Algorithm 3.1. If $I = \{i_j \mid 1 \leq j \leq f\}$, then*

$$S_H := \{s \cdot t_i \mid s \in S(V),\ i \in I\}$$

is a transversal for the right cosets of H in G. Here $S(V)$ is defined as in Lemma 2.3.

Proof. Let $g \in G$. By the choice of I, there exists some $h \in H$ and some $i \in I$ such that $gN = ht_iN = hNt_i$. Thus, $h^{-1}gt_i^{-1} \in N$ can be written as xs with $x \in \langle V \rangle = H \cap N$ and $s \in S(V)$ by Lemma 2.3. Hence, $g = (hx)st_i \in Hst_i$. So S_H represents all cosets. It remains to show that each coset is represented only once. So assume $s, s' \in S(V)$ and $i, k \in I$ such that $Hst_i = Hs't_k$. Then $Ht_iN = HNt_i = HNt_k = Ht_kN$ and the choice of I implies $i = k$. Hence, $s's^{-1} \in H \cap N$ and Lemma 2.3 shows $s = s'$. □

The above proof immediately gives rise to the following algorithm which solves the second problem of Theorem 1.1.

ALGORITHM 3.5. *Given $g \in G$, the algorithm returns $s \in S_H$ such that $Hg = Hs$.*

(1) *Using Algorithm (A), compute $1 \leq j \leq f$ such that $gN = t_jN$.*
(2) *From the preprocessing, get V, i_j and $\tilde{h}_j$.*
(3) *Call Algorithm (B) to write $\tilde{h}_j^{-1}gt_{i_j}^{-1} \in N$ as a word w in $\{x_1, \dots, x_r\}$.*
(4) *Using Algorithm 2.4, write w as vs' with v a word in V and $s' \in S(V)$.*
(5) *Write $s't_{i_j}$ as a word s in $\{g_1, \dots, g_e\}$ and return s.*

REMARK 3.6. The element $g \in G$ of the above algorithm lies in H if and only if $s = 1$ if and only if $i_j = 1$ and $s' = 1$. The latter condition can be easily checked since s' is given as a word in the basis $\{x_1, \dots, x_r\}$. Further, if $g \in H$, then $g = \tilde{h}_jv$. The element v is given as a word in V. If one keeps track of the substitutions made in Algorithm 2.1, one can express each element in V as a word in W. Since elements in W are given as words in $\{h_1, \dots, h_m\}$, one can thus write $g = \tilde{h}_jv$ as a word in the given generating set $\{h_1, \dots, h_m\}$ of H. This solves Problem 3 of Theorem 1.1. Unfortunately, writing the elements in V as words in W may produce words of exponential size as the following example shows.

EXAMPLE 3.7 [**1**, p. 66]. Let N be freely generated by x and y. Consider the two sequences

$$v_j = x^jyx^{1-j} \text{ for } j \in \mathbb{N}, \quad w_1 = xy \text{ and } w_i = \begin{cases} xw_{i-1}y & \text{if } i \text{ is even,} \\ xw_{i-1}x^{-1} & \text{if } i > 1 \text{ is odd.} \end{cases}$$

For $m \in \mathbb{N}$, let $W_m := \{w_i \mid 1 \leq i \leq m\}$ and $V_m := \{v_i \mid 1 \leq i \leq m\}$. Then W_m consists of m elements each of length of at most $2m$ as words in $\{x, y\}$. Further, V_m is normalised Nielsen reduced and by induction, it follows that $v_i = w_i(v_{i-1}v_{i-3}v_{i-5}\dots)^{-1}$ for all $i \geq 1$.

Thus, V_m and W_m generate the same subgroup of N and the length of v_m when written as a word in W_m is not polynomial in m. (The length grows faster than the Fibonacci sequence.)

Finally, for the proof of Theorem 1.2, it remains to analyse the running time of the above algorithms.

Proof of Theorem 1.2. Let $n = \|g\| + \sum_{i=1}^m \|h_i\| + \sum_{j=1}^f \|t_j\| + \sum_{k=1}^r \|x_k\|$. We first discuss Algorithm 3.3. The preprocessing step runs in polynomial time in c and the size of the input as seen in Remark 3.2. Further, the set V consists of at most $\#W \leq fm$ elements of length of (in the x_i) at most cfn. Thus, the computation of $\mathcal{T}(V)$ runs in time $O((cfn)^3 \cdot (fm)^2)$. The last step runs in constant time.
Now we discuss Algorithm 3.5. By assumption, the first step runs in polynomial time and so does the preprocessing. The length of $\tilde{h}_j$ as a word in $\{h_1, \dots, h_m\}$ is at most m. Thus, $\|\tilde{h}_j^{-1}gt_{i_j}^{-1}\| \leq (m+2)n$. Hence, the third step runs in polynomial time. Further, the length of w (and thus of s') as a word in $\{x_1, \dots, x_r\}$ is at most $c(m+1)n$. Thus, the fourth step runs in time $O((c(m+2)n)^2 \cdot fm)$. Finally, t_{i_j} is a word of length $\leq fm$ in

$\{h_1, \dots, h_m\}$. Since we assume that $\|x_i\| \leq n$ and $\|h_k\| \leq n$ for all i and k, the last step runs in polynomial time in n, m, f and c. □

4. Examples $\mathrm{PSL}_2(\mathbb{Z})$ **and** $\mathrm{SL}_2(\mathbb{Z})$**.** In this section, let G be either the special linear group $\mathrm{SL}_2(\mathbb{Z})$ or the modular group $\mathrm{PSL}_2(\mathbb{Z}) = \mathrm{SL}_2(\mathbb{Z})/\{\pm 1\}$. Further, let $S = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ and $U = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 1 \end{smallmatrix}\right)$ be elements in G.

We will show that Algorithms 1.1 and 1.2 can be applied to G, provided that the elements of G are given as words in S and U. For this, we have to give a finite index-free normal subgroup of G, and a transversal of this subgroup in G, as well as the corresponding algorithms (A) and (B).

Let o be the order of S in G, i.e. $o = 4$ if $G = \mathrm{SL}_2(\mathbb{Z})$ and $o = 2$ otherwise. Then,

$$\begin{aligned} \mathrm{SL}_2(\mathbb{Z}) &= \langle S \rangle *_{\{\pm I_2\}} \langle U \rangle \cong C_4 *_{C_2} C_6, \\ \mathrm{PSL}_2(\mathbb{Z}) &= \langle S \rangle * \langle U \rangle \cong C_2 * C_3 \end{aligned} \tag{1}$$

are amalgamated and free products respectively (see for example [**6**]). Thus, every element $g \in G$ can be written as a word in S and U. The length of such a word will be denoted by $\|g\|$. Moreover, from the above isomorphism it follows that every $g \in G$ can be written uniquely as a word

$$\tau(g) = S^{i_0} U^{i_1} S U^{i_2} S \dots \tag{2}$$

such that $0 \leq i_0 \leq o - 1$ and $i_k \in \{\pm 1\}$ for $k \geq 1$.

LEMMA 4.1. *The commutator subgroup G' of G is a free group of index*

$$[G : G'] = \begin{cases} 12 & \textit{if } G = \mathrm{SL}_2(\mathbb{Z}) \\ 6 & \textit{if } G = \mathrm{PSL}_2(\mathbb{Z}) \end{cases}$$

with basis $\{x := SUS^{-1}U^{-1},\ y := S^{-1}U^{-1}SU\}$.

Proof. The fact that G' is free is well known and the index $[G : G']$ follows immediately from equation (1). Suppose now $1 \neq g \in G'$ such that $g = \tau(g)$. Then a case by case discussion of the possible terminal words of g shows that there exists some $z \in \{x, y, x^{-1}, y^{-1}\}$ such that $\|\tau(gz)\| < \|g\|$. Thus, $\{x, y\}$ generates G'. □

- From the isomorphisms in equation (1) it follows that there exists an epimorphism $\varphi\colon G \to \mathbb{Z}/[G : G']\mathbb{Z}$ such that $\varphi(S) = 3$ and $\varphi(U) = 2$. The kernel of φ coincides with G'. Hence, a transversal of G' in G is given by

$$\{S^i U^j \mid 0 \leq i \leq o - 1,\ -1 \leq j \leq 1\}.$$

- Algorithm (A): Given $g \in G$ return $S^i U^j$, where $1 \leq i \leq o - 1$ and $-1 \leq j \leq 1$ such that $\varphi(g) = 3i + 2j \mod [G : G']$.
- Algorithm (B): Given $g \in G'$ as a word in S and U, the following algorithm writes g as a word w in x and y.
  (1) Initialise (g, w) by $(\tau(g), 1)$.
  (2) While $g \neq 1$, find $z \in \{x, x^{-1}, y, y^{-1}\}$ such that $\|\tau(gz)\| < \|g\|$ and replace (g, w) by $(\tau(gz), z^{-1}w)$.
  (3) Return w.

Note that Algorithm (B) terminates by the proof of Lemma 4.1.

It is clear that both algorithms run in time $O(\|g\|)$ and the word w returned by Algorithm (B) has at most length $\|g\|$. Hence, the algorithms of Theorem 1.1 are applicable to (P)$\mathrm{SL}_2(\mathbb{Z})$. Moreover, by Theorem 1.2, these algorithms run in polynomial time in the size of the input.

REMARK 4.2. Suppose H is a finitely generated subgroup of $\mathrm{PSL}_2(\mathbb{Z})$. Then Algorithm 3.3 can decide if the index $k := [\mathrm{PSL}_2(\mathbb{Z}) : H]$ is finite. If so, right multiplication of $\mathrm{PSL}_2(\mathbb{Z})$ on $H \setminus \mathrm{PSL}_2(\mathbb{Z})$ induces a homomorphism $\pi : \mathrm{PSL}_2(\mathbb{Z}) \to \mathrm{Sym}(k)$, where $\mathrm{Sym}(k)$ denotes the symmetric group on k letters. The images $\pi(S)$ and $\pi(U)$ can be worked out by $2k$ calls to Algorithm 3.5. Using Hsu's criterion [**3**, Theorem 3.1] one can then decide whether H is a congruence subgroup, i.e. whether there exists some $\ell \geq 2$ such that H contains the full kernel of the canonical epimorphism $\mathrm{PSL}_2(\mathbb{Z}) \to \mathrm{PSL}_2(\mathbb{Z}/\ell\mathbb{Z})$. One only has to check whether the permutations $\pi(S)$ and $\pi(U)$ satisfy a few short relations. This can be done in time $O(k)$.

REMARK 4.3. By Kurosh's subgroup theorem, every subgroup H of $\mathrm{PSL}_2(\mathbb{Z})$ is a free product $H_1 * H_2 * H_3$, where H_3 is free and H_1, H_2 are freely generated by elements of order 2 and 3 respectively. A constructive decomposition of H into free generators of H_i can be accomplished by using Nielsen reductions as in Algorithm 2.1 (see [5, Propositions 2.2 and 2.3). Note that in this case, the reduction operator ρ has to be replaced with τ from equation (2) and we use the lexicographical order satisfying $S <_l U <_l U^{-1}$.

REFERENCES

1. J. Avenhaus and K. Madlener, The Nielsen reduction and p-complete problems in free groups, *Theor. Comput. Sci.* **32** (1984), 61–76.

2. W. Bosma, J. Cannon and C. Playoust, The Magma algebra system. I. The user language, *J. Symb. Comput.* **24**(3–4) (1997), 235–265.

3. T. Hsu, Identifying congruence subgroups of the modular group, *Proc. Amer. Math. Soc.* **124**(5) (1996), 1351–1359.

4. A. Karrass and D. Solitar, On finitely generated subgroups of a free group, *Proc. Amer. Math. Soc.* **22**(1) (1969), 209–213.

5. R. C. Lyndon and P. E. Schupp, *Combinatorial group theory* (Springer, New York, NY, 1977).

6. J.-P. Serre, *Trees* (Springer, New York, NY, 1980).