SIAM J. Comput. © 2007 Society for Industrial and Applied Mathematics
Vol. 37, No. 2, pp. 425–459

MEMBERSHIP PROBLEM FOR THE MODULAR GROUP[*]

YURI GUREVICH[†] AND PAUL SCHUPP[‡]

Abstract. The modular group plays an important role in many branches of mathematics. We show that the membership problem for the modular group is decidable in polynomial time. To this end, we develop a new syllable-based version of the known subgroup-graph approach. The new approach can be used to prove additional results. We demonstrate this by using it to prove that the membership problem for a free group remains decidable in polynomial time when elements are written in a normal form with exponents.

Key words. $\mathrm{PSL}_2(\mathbb{Z})$, combinatorial group theory, folding, free group, geometric group theory, membership problem, modular group, polynomial time, syllabic presentation, unimodular matrices

AMS subject classifications. 03B25, 03D40, 11U05, 14L35, 15A30, 15A36, 20F10, 20G99

DOI. 10.1137/050643295

1. Introduction.

In this paper, a *unimodular matrix* is a 2×2 integer matrix with determinant 1. The multiplicative group of unimodular matrices is known as $\mathrm{SL}_2(\mathbb{Z})$, the special linear group of 2×2 matrices over the ring of integers. The modular group $\mathrm{PSL}_2(\mathbb{Z})$, the projective special linear group of 2×2 matrices over integers, is the quotient of the group $\mathrm{SL}_2(\mathbb{Z})$ modulo the congruence relation that equates a matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ with its negative $\left(\begin{smallmatrix} -a & -b \\ -c & -d \end{smallmatrix}\right)$. The modular group has numerous equivalent characterizations in various parts of mathematics [2, section 1]. In particular, it is the group of complex fractional linear transformations $z \mapsto \frac{az+b}{cz+d}$ with integer coefficients and $ad - bc = 1$.

Recall the membership problem for a group G: given elements $h_1, \ldots, h_n$ and w, determine whether w belongs to the subgroup H generated by $h_1, \ldots, h_n$. This presumes a fixed representation form for group elements. In the case of the modular group, group elements are represented by unimodular matrices. A matrix and its negative represent the same group element. The entries are written in the standard decimal notation. The *size* of an entry is the length of its decimal notation, and the *size* of a unimodular matrix is the sum of the sizes of the four entries.

Remark 1.1 (uniformity). The membership problem above is sometimes called uniform because the subgroup H is not fixed. The problem of deciding whether a given element w of the group G belongs to a fixed subgroup H is called the *membership problem for H in G*. We restrict attention to uniform membership problems and will not use the adjective "uniform."

The membership problem for the modular group, more exactly its bounded version, was raised by Gurevich [4], who was looking for a hard-on-average [6, 3] algebraic NP problem with a natural probability distribution on the instances. In the bounded version of the membership problem for a group G, in addition to a tuple $(h_1, \ldots, h_n, w)$, one is given a positive integer B in the unary notation; the question becomes whether w is a product of at most B of the elements h_i and their inverses.

[*]Received by the editors October 21, 2005; accepted for publication (in revised form) December 12, 2006; published electronically May 16, 2007.
http://www.siam.org/journals/sicomp/37-2/64329.html

[†]Microsoft Research, One Microsoft Way, Redmond, WA 98052 (gurevich@microsoft.com).
[‡]Mathematics, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (schupp@math.uiuc.edu).

After examining the bounded membership problem for the modular group (with an appropriate natural probability distribution), Gurevich conjectured that the problem is not hard on average.

In [2], Cai et al. proved that the bounded membership problem for the modular group is indeed polynomial time on average. They also proved that the unbounded membership problem for the modular group is polynomial time on average. Furthermore, consider the variant of the membership problem definition in which "subgroup" is replaced with "submonoid." The subgroup membership problem can be seen as a special case of the submonoid membership problem where the set $\{h_1, \ldots, h_m\}$ is closed under inverses. Cai et al. proved that both, bounded and unbounded, submonoid membership problems for the modular group are polynomial time on average [2, Theorem 1.1]. All their proofs are constructive: the desired decision procedures are exhibited. In this paper, the proofs are constructive as well.

Proviso 1.2. In this paper, all proofs of the existence of algorithms are constructive. The desired algorithms are exhibited.

As far as the worst-case analysis is concerned, Cai et al. established that the two submonoid membership problems are NP-hard. The bounded membership problem for the modular group was proved NP-hard in [1]. More precisely, it is the group $\mathrm{SL}_2(\mathbb{Z})$ that is called the modular group in [4, 1], and it is the bounded membership problem for $\mathrm{SL}_2(\mathbb{Z})$ that is proved NP-hard in [1]. But the same proof establishes also the NP-hardness of the bounded membership problem for $\mathrm{PSL}_2(\mathbb{Z})$.

THEOREM 1.3 (main). *The membership problem for the modular group is decidable in polynomial time.*

Group membership problems tend to be undecidable [7]. The modular group is atypical from that point of view. It is curious also that, in the case of the modular group, the unbounded membership problem is easier than the bounded one.

We do not try to optimize the decision algorithm and minimize its running time. This gives us freedom to ignore various details, most importantly the details related to various data structures. It is clear though that the algorithm is feasible.

To solve the membership problem for the modular group, we develop a new version of the subgroup-graph approach of combinatorial group theory. The subgroup-graph approach, also known as the folding method, was originated by Stallings [10]. It was employed and developed further in particular by Kapovich and Miasnikov [5] and Schupp [9]. Our version of the approach is combinatorial. The closest version in the literature is that of Kapovich and Miasnikov [5].

The subgroup graph in question is really a finite automaton, in general nondeterministic. We call it a subgroup recognizer or simply a recognizer. We call our approach the syllabic recognizer approach or simply the syllabic approach. It is based on the notion of a syllable. A recognizer reads words one syllable at a time. Another distinctive feature of our approach is that we fold paths rather than edges.

In section 2, we introduce syllabic representations of abstract groups and explain the basics of the syllabic approach. A syllabic presentation of an abstract group G with a fixed finite set of generators is given by means of four items. First, there is a finite alphabet with letters representing the generators and possibly some auxiliary symbols. Second, there is a set of strings in the given alphabet. These strings are called syllables. Finite concatenations of syllables are called words. The words with the concatenation operation form a semigroup. Third, there is an involution on the syllables called the inverse operation. It extends to words in the obvious way. Finally, there is a congruence relation on the word semigroup with the inverse operation such that the quotient algebra is isomorphic to G. Notice that words are strings in the given alphabet. Accordingly the size of a word is the length of the string. Often it suffices to describe the syllables, and the rest of the syllabic representation becomes obvious.

Example 1.4 (standard and succinct free groups). Consider the case in which G is a free group and the fixed set of generators consists of the free generators of G. The standard representation of G is obtained when the syllables have the form a or a^{-1}, where a is a free generator. Another, exponentially more succinct representation of G is obtained when the syllables have the form a^i, where a is a free generator and i is a nonzero integer in decimal notation. For brevity we will speak about *standard free groups* and *succinct free groups* meaning free groups in the standard representation and free groups in the succinct representation, respectively.

Proviso 1.5 (the free group). We will study finitely generated free groups with at least two free generators. The number of free generators will play an insignificant role. To simplify terminology, we fix some integer ≥ 2 and restrict attention to that particular free group.

The membership problem for the standard free group is decidable in polynomial time [8, 7]. We construct a new decision algorithm for the problem in section 2. The purpose of this is twofold: to illustrate the syllabic approach on a simple example and to produce a proof template for sections 3–5.

In section 3 we prove that the membership problem for the free group remains feasible when we go from the standard representation to the succinct.

THEOREM 1.6 (succinct free group). *The membership problem for the succinct free group is decidable in polynomial time.*

But the proof of polynomial time decidability is much harder in the case of the succinct representation. One reason for this is that the number of syllables is infinite. Another reason is that the classical edge-folding technique used in the standard case is utterly inadequate in the succinct case. Instead we have to fold paths.

What has all this to do with the membership problem for the modular group? The bridge is the following well-known fact [8, section 1.4, Exercises 18–24]. Recall that, in the notation of combinatorial group theory, $\langle g \mid g^n \rangle$ is a cyclic group of order n with generator g.

PROPOSITION 1.7 (modular group as a free product). *The modular group is isomorphic to the free product* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$.

Example 1.8 (standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ and succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$). We give two syllabic representations of the group $G = \langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ with the set s, t of generators. (The symbols s, t allude to "second" and "third," respectively.) The standard representation of G is obtained when the syllables are s, t, t^{-1}. Another, exponentially more succinct representation of G is obtained when the syllables are s, $(ts)^n t$, and $(t^{-1}s)^n t^{-1}$, where n is a natural number in decimal notation. For brevity we will speak about the *standard* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ and the *succinct* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ meaning $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ in the standard representation and $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ in the succinct representation, respectively.

In section 4, we prove that the membership problem for the standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ is decidable in polynomial time. The proof sheds some light on the membership problem for the modular group but is not too useful all by itself.

Ideas and the terminology of sections 2–4 are used in the crucial section 5, where we study the succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$.

THEOREM 1.9 (succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$). *The membership problem for the succinct* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ *is decidable in polynomial time.*

The proof closely follows that of Theorem 1.6, but new difficulties arise. The main new difficulty is related to the fact that the role of free generators is played by the string ts. This fake free generator is not atomic: it is the concatenation of t and s. The atomicity of free generators was implicitly used in the proof of Theorem 1.6.

The main theorem is finally proved in section 6 where we give a polynomial time reduction of the membership problem for the modular group to that for the succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$. That reduction is relatively simple and is essentially independent from the preceding sections.

The syllabic approach can be used to prove additional results. But this is a topic for separate papers. Here we focus on the modular group.

The intended audience for this paper is computer scientists rather than group theorists. Accordingly we do not presume the knowledge of group theory.

2. Syllabic recognizer approach: Basics and illustration.

We explain the basics of the syllabic recognizer approach and illustrate the approach by constructing a new decision algorithm for the membership problem for the free group in standard presentation. The construction is used as a template for generalization in sections 3–5.

2.1. Syllabic presentations of groups.

The peculiarity of combinatorial/geometric group theory, in comparison to abstract group theory, is that one has to deal not only with group elements but also with their representations. The standard group presentation form employs generators and relators [8, 7]. We introduce a new group presentation form.

Definition 2.1 (syllabic group presentation). *Let G be a group with a fixed finite set of generators. A syllabic presentation of G with fixed generators is given by means of four items: a finite alphabet, a set of syllables, an involution on syllables called the inverse operation, and an equality relation on the semigroup of words built from the syllables. We describe the four items in greater detail.*

- *Alphabet: The symbols of the alphabet split into two categories. First, there are letters that represent the fixed generators, one letter per generator. We call them original letters. We call the remaining symbols auxiliary. The symbols of the alphabet are linearly ordered.*
- *Syllables: Syllables are strings in the alphabet described above. We reserve the Greek letter σ to denote syllables. The number of syllables may be finite or infinite. Finite concatenations of syllables are called words. The empty string 1 is a word. The words with the concatenation operation form a semigroup called the word semigroup. It is in fact a monoid.*
- *Inverse operation: The inverse operation takes a syllable σ to another syllable σ^{-1}. It is an involution, so that $(\sigma^{-1})^{-1} = \sigma$. We extend the inverse operation to words in the obvious way: $(\sigma_1\sigma_2 \ldots \sigma_k)^{-1} = \sigma_k^{-1} \ldots \sigma_2^{-1}\sigma_1^{-1}$.*
- *Equality: The equality relation is a congruence of the word semigroup with the inverse operation. In other words, it respects the concatenation operation as well as the inverse operation:*

$$(x_1 = y_1) \ \wedge \ (x_2 = y_2) \ \longrightarrow \ x_1y_1 = x_2y_2,$$
$$x_1 = y_1 \ \longrightarrow \ x_1^{-1} = y_1^{-1}$$

  for all words x_1, x_2, y_1, y_2. It is required that the quotient algebra is isomorphic to G. So two words are equal if they denote the same element of G.

Finally, notice that words are strings in the given alphabet. Accordingly the size, *or* length, *of a word is the number of (the occurrences of) alphabet symbols in the word.*

Remark 2.2 (equality vs. identity). Thus we have two competing relations on words. One is the identity of words as strings in the given alphabet, and the other is the equality of words (as names for the group elements). Notice that we saw the distinction between identity and equality already, in the very beginning of the Introduction, when we recalled the definition of $\mathrm{PSL}_2(\mathbb{Z})$. Elements of $\mathrm{PSL}_2(\mathbb{Z})$ are modular matrices, but every matrix is equated with its negative. For our purposes, it is convenient to use an equality relation that is different from the identity relation. We use a word as a name for a group element, and we don't have to mention the corresponding equivalence class of words all the time. But there is an obvious awkwardness in calling distinct words equal, and so we will have to be careful to avoid a confusion.

In the rest of this subsection we consider the free group in the standard presentation.

DEFINITION 2.3 (standard free group). *The* standard free group *is the free group in the following syllabic presentation.*

- *Alphabet: The* alphabet *consists of original letters denoting the free generators and one auxiliary symbol* $^{-1}$. *The alphabet is linearly ordered in some way; it will play no role what order it is exactly.*
- *Syllables:* Syllables *are strings of the form* a *or* a^{-1}, *where* a *is an original letter and* $^{-1}$ *is the auxiliary symbol.*
- *Inverses: Syllables* a *and* a^{-1} *are the inverses of each other.*
- *Equality: It is the least congruence for the word semigroup with the inverse operation such that* $xx^{-1} \sim 1$ *for every word* x.

It is easy to see that the quotient of the word semigroup over the equality relation is indeed isomorphic to the free group.

A combinatorial characterization of the equality relation. If a word $x = x_1\sigma\sigma^{-1}x_2$, we say that the word x_1x_2 is obtained from x by a single cancellation. (Recall that σ ranges over syllables.) It is easy to see that words w_1 and w_2 are equal if and only if there is a sequence of words $x_0, \ldots, x_\ell$ such that x_0 is w_1, x_ℓ is w_2, and for every pair x_i, x_{i+1} of successive words, one is obtained from the other by a single cancellation.

DEFINITION 2.4 (reduced words). *A word* w *is* reduced *if no successive syllables of* w *form an inverse pair.*

LEMMA 2.5 (word reduction).

1. *Every word* w *is equal to a unique reduced word. That reduced word is called the* reduct *of* w.
2. *There is a polynomial-time* word reduction algorithm *that transforms any word* w *to its reduct.*

These facts are well known [8, 7] and relatively easy to verify.

2.2. Recognizers and the construction algorithm.

2.2.1. Recognizers: Definitions.

DEFINITION 2.6 (recognizer). *A* recognizer R *is a nondeterministic finite state automaton with the input alphabet* Σ *subject to the following conditions:*

- *Numerical states: The states are positive integers.*
- *Finiteness:* R *has only finitely many transitions.*
- *Reversibility: For every transition* (u, σ, v), *from state* u *on input symbol* σ *to state* v, *there is an inverse transition* (v, σ^{-1}, u).
- *Cyclicity: The initial state is the only final (or accepting) state.*
- *Connectivity: For every state* u, *there is a string* s *of input symbols such that the computation of* R *on* s *ends at* u.

Remark 2.7 (numerical states). What objects can serve as states of a recognizer? Algebraically speaking, the nature of the states is of no importance. Algorithmically speaking, it is important to have a reasonable representation of the states. We will use in particular the fact that the states are linearly ordered.

DEFINITION 2.8 (recognizer's size). *The* size *of a positive integer n is the size of the standard decimal representation of n. The* size *of a transition (u, σ, v) is the sum of the sizes of the three components. The* size *of the recognizer is the sum of the sizes of its transitions.*

Intuitively the size of the recognizer is the number of characters in a reasonable representation of recognizers. The fact that we care only that our algorithms are polynomial time gives us a large freedom in defining the sizes. We could have used the unary notation for the states.

DEFINITION 2.9 (recognizer's subgroup). *The initial state is also called the* origin *and is denoted o. The language recognized by R, that is, the set of words accepted by R, is denoted $L(R)$. It is easy to see that $L(R)$ is closed under concatenation and under the inverse operation. Define $\Gamma(R)$ to be the set of group elements w such that the word w is in $L(R)$. It is easy to see that $\Gamma(R)$ is a subgroup of G. We will say that R* recognizes *$\Gamma(R)$.*

DEFINITION 2.10 (equivalence of recognizers). *Recognizers R_1 and R_2 are* equivalent *if $\Gamma(R_1) = \Gamma(R_2)$.*

2.2.2. A graph-theoretic view of recognizers. A *recognizer* R over Σ can be viewed as a directed graph with a distinguished vertex o, the *origin*, where every edge is labeled with an element of Σ and parallel edges are allowed. If e is an edge with start-vertex u, end-vertex v, and label σ, then the triple (u, σ, v) is the *profile* of e. It is required that

- R has finitely many vertices and edges;
- R is strongly connected so that there is a path from any vertex u to any other vertex v;
- different edges have different profiles so that the profile of an edge identifies it uniquely; and
- for every edge $e = (u, \sigma, v)$, there is an inverse edge $e^{-1} = (v, \sigma^{-1}, u)$.

We will use both the automata-theoretic terminology as well as the graph-theoretic terminology. Some graph-theoretic terms are used in different ways by different authors. To fix some graph-theoretic terminology, we give the following definitions.

DEFINITION 2.11 (paths). *A* path *is a sequence $\langle e_1, e_2, \ldots, e_\ell \rangle$ of edges such that the start-vertex of e_{i+1} is the end-vertex of e_i. We do not distinguish between an edge e and the single-edge path formed by e. Consider a path*

$$\pi = \langle (u_0, \sigma_1, u_1), (u_1, \sigma_2, u_2), \ldots, (u_{\ell-1}, \sigma_\ell, u_\ell) \rangle.$$

The vertex sequence *of π is $\langle u_0, u_1, \ldots, u_\ell \rangle$. The vertices u_i with $0 < i < \ell$ are* internal*. The string $\sigma_1\sigma_2 \ldots \sigma_\ell$ is the* label *of π, and the triple $(u_0, \sigma_1\sigma_2 \ldots \sigma_\ell, u_\ell)$ is the* profile *of π. We use the $+$ sign to indicate the concatenation of paths; if a path π_1 has 7 edges and a path π_2 has 11 edges, then the path $\pi_1 + \pi_2$ has 18 edges.*

While every edge is uniquely determined by its profile, this is not necessarily true for paths. For example, you may have different paths $\langle (u_0, aa, u_1), (u_1, a, u_2) \rangle$ and $\langle (u_0, a, v), (v, aa, u_2) \rangle$ with the profile (u_0, aaa, u_2).

DEFINITION 2.12 (branches, cycles, and nooses). *Consider a path $\pi = \langle e_1, \ldots, e_\ell \rangle$ with vertex sequence $\langle u_0, u_1, \ldots, u_\ell \rangle$, and assume that the nonfinal vertices $u_0, u_1, \ldots, u_{\ell-1}$ of π are all distinct.*

- *π is a* branch *if the final vertex u_ℓ differs from all nonfinal ones.*
- *π is a* cycle at u_0 *if the final vertex coincides with the start-vertex u_0.*
- *π is a* noose *if the final vertex coincides with an internal vertex u_i.*

If π is a noose and $u_\ell = u_i$, then π splits into the loop $\langle e_{i+1}, \dots, e_\ell \rangle$ *and the* tail $\langle e_1, \dots, e_i \rangle$ *of the noose.*

Definition 2.13 (path segments). *If u, v are vertices on a path π and if there is a contiguous segment of π with initial vertex u and final vertex v, then $\pi[u, v]$ is the shortest of such segments.*

Definition 2.14 (disjoint paths). *Paths π_0 and π_1 are* internally disjoint *if no internal vertex of π_i occurs on π_{1-i}. Further, paths π_0 and π_1 are* disjoint off a vertex set U *if all vertices that occur on both paths belong to U. If π_0 and π_1 are disjoint off $\{\nu\}$, we say that they are* disjoint off the vertex ν.

2.2.3. Construction algorithm.

Lemma 2.15 (construction). *There exists a polynomial-time* construction algorithm *that, given arbitrary words $h_1, h_2, \dots, h_m$, constructs a recognizer R such that $\Gamma(R)$ is the subgroup generated by the group elements $h_1, \dots, h_m$.*

Proof. The desired recognizer R is a bouquet of m cycles labeled with $h_1, \dots, h_m$ and having only the initial state o in common. The desired algorithm is this. Start with a naked origin vertex o. For each generator h_j, put a cycle C_j with profile (o, h_j, o) around o. If $h_j = \sigma_1 \dots \sigma_n$ and $C_j = \langle (u_0, \sigma, u_1), \dots, (u_{n-1}, \sigma_n, u_n) \rangle$, where $u_0 = u_n = o$ and the internal vertices are new, then for each existing edge (u, σ, v), create an inverse edge (v, σ^{-1}, u). That completes the construction of R.

It is easy to see that $L(R)$ is the least set of words that contains $h_1, h_2, \dots, h_m$ and is closed under concatenation and the inverse operation. □

Example 2.16. Suppose that G is the standard free group, a, b, and c are free generators of G, $m = 2$, $h_1 = ab$, and $h_2 = c^{-1}b$. Then R consists of three vertices and eight edges. There is a unique a-labeled edge from o to a nonorigin vertex u. Call the other nonorigin vertex v. Then the edges are

$$(o, a, u), (u, b, o), (o, c^{-1}, v), (v, b, o)$$

and the inverses of these four edges. R accepts any concatenation $x_1 x_2 \dots x_\ell$, where every $x_i \in \{h_1, h_2, h_1^{-1}, h_2^{-1}\}$. In particular, R accepts $(ab)(c^{-1}b)^{-1} = ac$.

Remark 2.17 (nondeterministic algorithms). When we claim that there is a polynomial-time algorithm, we mean a deterministic algorithm, of course. But the algorithm described in the proof of the construction lemma is nondeterministic, and so the description is incomplete. What is missing is how to determinize the construction described in the proof. This easy task is left to the reader. The inessential nondeterminism of that sort allows us to simplify proofs and will be used over and over again.

2.3. Membership criterion and the reading algorithm for the standard free group. In this section, we deal only with the standard free group. In particular the notion of fat will be redefined again and again as we deal with other syllabically presented groups.

Definition 2.18 (fat). *Let R be a recognizer. For every syllable σ and every vertex v,* $\mathrm{Fat}_R(\sigma, v) = \max(0, i-1)$*, where i is the number of σ-edges from v. The subscript is omitted when the context uniquely defines the recognizer.*

A recognizer is deterministic if and only if every $\mathrm{Fat}(\sigma, v) = 0$.

Remark 2.19 (deterministic recognizers). When is a nondeterministic finite state automaton deterministic? There are two common definitions in the literature. The stricter definition requires that, for every label and every state u, there is exactly one transition with that label from that state. The more liberal definition requires only that, for every label and every state, there is at most one transition with that label from that state. We adopt the more liberal definition.

LEMMA 2.20 (membership criterion). *Let R be a deterministic recognizer and w a word. The group element w belongs to $\Gamma(R)$ if and only if R accepts the reduct of w.*

Proof. If R accepts the reduct of w, then by the recognizer's subgroup definition in section 2.2, the group element w belongs to $\Gamma(R)$.

Now suppose that the group element w belongs to $\Gamma(R)$ so that R accepts at least one word equal to w. Consider a shortest word w_0 equal to w and accepted by R, and let π be the accepting run. The word w_0 is reduced. Otherwise π has the form

$$\pi_1 + \langle (u_1, \sigma, v), (v, \sigma^{-1}, u_2) \rangle + \pi_2.$$

Since $\mathrm{Fat}(\sigma^{-1}, v) = 0$, we have $u_1 = u_2$. Accordingly $\pi_1 + \pi_2$ is a run that accepts a word equal to w that is shorter than w_0, which contradicts the choice of w_0. ☐

LEMMA 2.21 (reading). *There is a polynomial-time algorithm (called the* reading algorithm *below) that, given a deterministic recognizer R and a word w, determines whether the group element $w \in \Gamma(R)$.*

Proof. Use the word reduction algorithm of section 2.1 to compute the reduct w_0 of w. By the membership criterion in section 2.3, $\Gamma(R)$ contains the group element w if and only if R accepts w_0.

To determine whether R accepts w_0, run R on w_0. Let $w_0 = \sigma_1 \ldots \sigma_\ell$ and $v_0 = o$. If, after reading an initial segment $\sigma_1 \ldots \sigma_i$ of w_0, R arrives to a state v_i without an outgoing edge labeled with σ_{i+1}, then R rejects w_0. If R reads all of w_0 and winds up at a vertex v_ℓ, then R accepts w_0 if and only v_ℓ is o. ☐

In the remaining part of this section, we construct a polynomial-time algorithm that transforms any recognizer to an equivalent deterministic recognizer.

2.4. Vertex identification and edge folding. We return to consider the general situation. Recall that if H is a subgroup of G and $g \in G$, then the set $Hg = \{hg \ : \ h \in H\}$ is a *(right) coset* of the subgroup H.

LEMMA 2.22. *Let R be a recognizer and $H = \Gamma(R)$. For every two paths (o, x, v) and (o, y, v) from the origin to the vertex v, $Hx = Hy$.*

Proof. The concatenation of (o, x, v) and (v, y^{-1}, o) is an accepting run on xy^{-1}, so $xy^{-1} \in H$ and $Hx = Hy$. ☐

The lemma and the fact that, for every v, there is a path from o to v lead us to the following definition.

DEFINITION 2.23 (vertex's coset). *Let R be a recognizer R and $H = \Gamma(R)$ and v be a vertex of G. Coset(v) is the set Hw, where w is the label of any path from o to v.*

LEMMA 2.24 (coset stability). *For any path (u, w, v),* $\mathrm{Coset}(v) = (\mathrm{Coset}(u))w$.

Proof. Let π be a path (o, x, u) and ρ be the given path (u, w, v). Then $\pi + \rho$ is a path with profile (o, xw, v). We have $\mathrm{Coset}(v) = H(xw) = (Hx)w = (\mathrm{Coset}(u))w$. ☐

Example 2.16 (continuation). Because of paths (o, a, u) and (u, b, o), we have $\mathrm{Coset}(u) = Ha = Hb^{-1}$. Because of paths (o, c^{-1}, v) and (v, b, o), we have $\mathrm{Coset}(v) = Hc^{-1} = Hb^{-1}$. In particular $\mathrm{Coset}(u) = \mathrm{Coset}(v)$.

DEFINITION 2.25 (morphisms). *Let R, S be recognizers with vertex sets U, V, respectively, operating on the same input alphabet Σ. A* morphism *from R to S is a function $\mu : \ U \to V$ satisfying the following conditions.*

- *$V = \{\mu(u) : \ u \in U\}$, and $\mu(o_R) = o_S$.*
- *S has an edge (v_1, σ, v_2) if and only if R has an edge (u_1, σ, u_2) with $\mu(u_1) = v_1$ and $\mu(u_2) = v_2$.*
- *If $\mu(u_1) = \mu(u_2)$, then* $\text{Coset}(u_1) = \text{Coset}(u_2)$.

LEMMA 2.26 (morphism lemma). *If there is a morphism from a recognizer R to a recognizer S, then $\Gamma(R) = \Gamma(S)$.*

Proof. Let $H = \Gamma(R)$, $I = \Gamma(S)$, and μ be a morphism from R to S. To simplify notation, $\mu(u)$ is denoted u'. Any accepting run (o, w, o) of R gives rise to an accepting run (o', w, o') of S. Thus $L(R) \subseteq L(S)$ and $H \subseteq I$. Furthermore, any run (o, w, u) of R gives rise to an accepting run (o', w, u') of S. Thus,

$$\text{Coset}(u) = Hw \implies \text{Coset}(u') = Iw.$$

To prove the other inclusion, it suffices to establish the opposite implication:

$$\text{Coset}(v') = Iw \implies \text{Coset}(v) = Hw.$$

Indeed suppose that $w \in I$. Then $\text{Coset}(o') = Iw$. Hence $H = \text{Coset}(o) = Hw$ and $w \in H$.

We prove the implication by induction on the number ℓ of edges in a shortest path π from o' to v'. The case $\ell = 0$ is trivial. Suppose that $\ell > 0$ so that the label of π has the form $x\sigma$ and $\text{Coset}(v') = Ix\sigma$. We prove that $\text{Coset}(v) = Hx\sigma$. Since μ is a morphism, the penultimate vertex on π has the form u' for some R-vertex u. We have $\text{Coset}(u') = Ix$ and, by the induction hypothesis, $\text{Coset}(u) = Hx$. Since μ is a morphism, there is an edge (u_0, σ, v_0) with $u_0' = u'$ and $v_0' = v'$. We have

$$\begin{aligned}&\text{Coset}(v) = \text{Coset}(v_0) = (\text{Coset}(u_0))\sigma \\ &= (\text{Coset}(u))\sigma = (Hx)\sigma = H(x\sigma).\end{aligned}$$

The first equality is by the morphisms definition, and the second equality is by the coset stability lemma. ◻

DEFINITION 2.27 (vertex identification). *Let R be a recognizer with vertices U, and let v_1, v_2 be two vertices of R such that* $\text{Coset}(v_1) = \text{Coset}(v_2)$. *To* identify *$v_1$ and v_2 in R means to construct a recognizer S with vertices $(U - \{v_1, v_2\}) \cup \{v\}$, where v is different from any vertex in $(U - \{v_1, v_2\})$, such that the map*

$$\mu(u) = \begin{cases} v & \textit{if } u = v_1 \textit{ or } v = v_2, \\ u & \textit{otherwise} \end{cases}$$

is a morphism from R to S.

It is easy to see that the desired S exists and is unique up to isomorphism. Let σ be any label u, be any vertex in $(U - \{v_1, v_2\})$, and i, j range over $\{1, 2\}$. S has an edge (u, σ, v) if and only if R has an edge of the form (u, σ, v_i). S has an edge (v, σ, u) if and only if R has an edge of the form (v_i, σ, u). And S has a loop v, σ, v if and only if R has an edge of the form (v_i, σ, v_j).

Remark 2.28 (vertex identification). There are several ways to implement S. One can remove v_1, v_2 and replace them with a fresh vertex v. One can remove one of vertices v_1, v_2 and use the remaining one as v. Our preferred way to implement S is this: view v_1 and v_2 as two names of the same vertex in S.

LEMMA 2.29 (vertex identification). *The identification of vertices u, v such that* $\text{Coset}(u) = \text{Coset}(v)$ *produces an equivalent recognizer.*

Proof. Use the vertex identification claim and the morphism lemma. □

Example 2.16 (continuation). Consider a recognizer S with two vertices o, v' and six edges:

$$(o,a,v'),(v',a^{-1},o),(o,b^{-1},v'),(v',b,o),(o,c^{-1},v'),(v',c,o).$$

Since $\text{Coset}(u) = \text{Coset}(v)$, there is a homomorphism μ of R onto D_1 such that $\mu(u) = \mu(v) = v'$. By the previous lemma, S is a recognizer for H. Note that S accepts ac while R does not.

Full vertex identification, when all vertices with the same coset are identified, reduces any recognizer R to a recognizer S where distinct vertices have different cosets. By the morphism lemma, $\Gamma(S) = \Gamma(R)$. Further, S is deterministic. Indeed suppose that we have edges (u,σ,v_1) and (u,σ,v_2) in S. By the coset stability lemma, $\text{Coset}(v_1) = (\text{Coset}(u))\sigma = \text{Coset}(v_2)$, and so $v_1 = v_2$. But notice that the question of whether Hw_1 is equal to Hw_2 is equivalent to the question of whether $w_1w_2^{-1} \in H$ which is an instance of the membership problem, the problem we want to solve in the first place. We need to get around this difficulty.

If R is a recognizer and $(u,a,v_1),(u,a,v_2)$ are edges in R, then by the coset stability lemma, $\text{Coset}(v_1) = (\text{Coset}(u))a = \text{Coset}(v_2)$ in R. This justifies the following definition.

Definition 2.30 (edge folding). *Let (u,σ,v_1) and (u,σ,v_2) be edges in a recognizer. To* fold edges (u,σ,v_1) and (u,σ,v_2) *is to identify the vertices v_1 and v_2.*

Lemma 2.31 (edge folding). *Folding edges does not change the recognizer subgroup.*

Proof. Use the vertex identification lemma. □

Example 2.16 (continuation). S is obtained from R by folding edges (o,b^{-1},u) and (o,b^{-1},v) together.

2.5. Fat reduction algorithm for the standard free group.

In this section, we deal exclusively with the standard free group.

Lemma 2.32 (fat reduction). *There is a polynomial-time* fat reduction algorithm *that transforms any recognizer R into an equivalent deterministic recognizer.*

Proof. The desired algorithm repeatedly folds distinct edges of the current recognizer with the same start-vertex u and the same label a until the recognizer becomes deterministic. By the edge-folding lemma, the recognizer subgroup does not change. It is easy to see that the algorithm works in polynomial time. □

The question arises whether the reducer makes all possible vertex identifications.

Lemma 2.33. *Let R be a deterministic recognizer for H. If (o,x,u) and (o,y,v) are paths with $Hx = Hy$, then $u = v$.*

It follows that distinct vertices of R have distinct cosets.

Proof. Induction on $n = |x| + |y|$. The basis of induction, when $n = 0$, is trivial. Assume that $n > 0$. Without loss of generality, x and y are reduced. Indeed, suppose that one of them, say x, is not reduced. Then x has the form $x_1\sigma\sigma^{-1}x_2$. By the determinacy of R, there is a path (o,x_1x_2,u). We have $Hx_1x_2 = Hx = Hy$. By the induction hypothesis, $u = v$.

If $x = 1$ so that $u = o$, then $Hy = H1 = H$ so that the group element y belongs to H. By the membership criterion, R accepts y, and so $v = o = u$. The case $y = 1$ is similar. Thus we can assume that neither word is empty.

Let $x = x'\sigma, y = y'\tau$, and let u', v' be the penultimate vertices in the paths (o,x,u) and (o,y,v), respectively. If $\sigma = \tau$, then $Hx' = Hx\sigma^{-1} = Hy\sigma^{-1} = Hy\tau^{-1} = Hy'$, and so $u' = v'$. Since R is deterministic, $u = v$. So we may assume that $\sigma \neq \tau$. Then the word xy^{-1} is reduced. Also $xy^{-1} \in H$ because $Hx = Hy$. So there is a cycle at o with label xy^{-1}. Since R is deterministic, there is a path (u, y^{-1}, o), and therefore there is a path (o, y, u). Since R is deterministic, the path (o, y, u) coincides with the path (o, y, v), and so $u = v$. $\square$

2.6. Membership problem for the standard free group.

THEOREM 2.34. *There is a polynomial-time decision algorithm for the membership problem for the free group G. More explicitly, there is an algorithm such that*

(i) *given words $h_1, \dots, h_m$ and w representing elements of G, the algorithm decides whether the subgroup H generated by $h_1, \dots, h_m$ contains w, and*

(ii) *the algorithm runs in time polynomial in $|h_1| + \cdots + |h_m| + |w|$.*

Proof. Use the construction algorithm of section 2.2 to construct a recognizer R_1 for H. Use the fat reduction algorithm of section 2.5 to transform R_1 into an equivalent deterministic recognizer R_2 for H. Finally, use the reading algorithm of section 2.3 to check whether $w \in H$. Since the three algorithms are polynomial time, the decision algorithm is polynomial time as well. $\square$

3. Succinct free group.

We introduce an exponentially more succinct representation of the elements of the free group and show that the membership problem remains polynomial-time decidable.

3.1. Succinct free group: Definition and word reduction.

DEFINITION 3.1 (succinct free group). *The* succinct free group *is the free group in the following syllabic presentation.*

- *Alphabet: The alphabet consists of original letters denoting the free generators and* 11 *auxiliary symbols* ^{0}, ^{1}, ..., ^{9}, *and* $^-$. *The alphabet is linearly ordered in some way; it will play no role in what order it is exactly.*
- *Syllables: Syllables are strings of the form a^i, where a is an original letter and i is a nonzero integer in decimal notation.*
- *Inverses: Syllables a^i and a^j are the inverses of each other if $i + j = 0$. To distinguish the representation of the free group from the standard representation, the new words will be called* exponent words, *and the old words will be called* unary words. *Any exponent word w expands in the obvious way to a unary word called the* unary expansion *of w. For example, $a_1^{-3}a_2^5$ expands to $a_1^{-1}a_1^{-1}a_1^{-1}a_2a_2a_2a_2a_2$.*
- *Equality: Exponent words are* equal *if their unary expansions are equal in the standard free group.*

Obviously the quotient of the word semigroup over the equality relation is isomorphic to the free group.

DEFINITION 3.2 (reduced exponent words). *An exponent word $w = a_1^{p_1} a_2^{p_2} \cdots a_k^{p_k}$ is* reduced *if every a_{i+1} differs from a_i.*

LEMMA 3.3 (exponent word reduction).

1. *Every exponent word w is equal to a unique reduced exponent word. That reduced exponent word is called the* reduct *of w.*
2. *There is a polynomial-time* exponent word reduction algorithm *that transforms any word w to its reduct.*

Proof. We describe the desired algorithm. If there are neighboring syllables a^i, a^j with the same base a, do the following. If $j = -i$, then remove the substring $a^i a^j$; otherwise, replace the substring $a^i a^j$ with the syllable a^{i+j}. Keep doing this until the exponent word is reduced. $\square$

3.2. Vertex creation and membership criterion.

DEFINITION 3.4 (edges). *An edge e of the form (u, a^i, v) is an* a-edge. *We assign to e the sign of i, so that e is positive (respectively, negative) if i is so. The* length *of e is the number of syllables in the unary expansion of its label a^i, so that the length of an edge is exponentially larger than the size of its label.*

DEFINITION 3.5 (edge splitting). *We define how to* split an edge $e = (u_1, x, u_2)$ *into two edges of lengths n_1 and n_2, respectively. It is presumed that n_1 and n_2 are positive integers such that $n_1 + n_2 = n$. Let x_1 be the prefix of x of length n_1 and let x_2 be the corresponding suffix. Add a new vertex v and edges*

$$(u_1, x_1, v), (v, x_1^{-1}, u_1), (v, x_2, u_2), (u_2, x_2^{-1}, v),$$

and remove edges e and e^{-1}.

DEFINITION 3.6 (vertex creation). *We define how to* create a new vertex *on a path $\pi = \langle e_1, \dots, e_i\rangle$ at distance n from the initial vertex u_0. It is presumed that*

$$\text{Length}(\langle e_1, \dots, e_i\rangle) < n < \text{Length}(\langle e_1, \dots, e_{i+1}\rangle)$$

for some i. Split e_{i+1} into two edges of lengths $n - \text{Length}(\langle e_1, \dots, e_i\rangle)$ and $\text{Length}(\langle e_1, \dots, e_{i+1}\rangle) - n$.

LEMMA 3.7 (vertex creation). *The creation of a new vertex preserves the recognizer subgroup and the amount of fat.*

Proof. It suffices to prove that splitting an edge preserves the recognizer subgroup. Suppose that R is the given recognizer and a new recognizer S is obtained from R by splitting an edge e of R into edges e_1 and e_2. Note that $\langle e_1, e_2\rangle$ is a path with the profile of e. The amount of fat at the new vertex is 0, and the amount of fat at any old vertex does not change, so $\text{Fat}(R) = \text{Fat}(S)$.

If $w \in L(R)$ and π is an accepting run of R on w, replace every occurrence of e in π with $\langle e_1, e_2\rangle$, and replace every occurrence of e^{-1} in π with $\langle e_2^{-1}, e_1^{-1}\rangle$. The result is an accepting run of S on a word equal to w. Suppose that $w \in L(S)$, and let π be an accepting run of S on w. The new vertex can occur only in the context $\langle e_1, e_2\rangle$, which can be replaced by e, or in the context $\langle e_2^{-1}, e_1^{-1}\rangle$, which can be replaced by e^{-1}. The result of the replacements is an accepting run of R on a word equal to w. ∎

DEFINITION 3.8 (paths). *The* length *of a path π is the sum of the lengths of its edges. For every original letter a, an* a-path *is a nonempty path composed of a-edges. A* positive a-path *is composed of positive a-edges, and a* negative a-path *is composed of negative a-edges. A* partisan a-path *is a positive or negative a-path.*

In addition to the standard notion of acceptance, will we need a more liberal one.

DEFINITION 3.9 (quasi transitions). *Suppose that π is a path with profile (u, x, v). If x is equal to a syllable or to 1, then π is a* quasi transition *and the syllable or the empty word 1 is the* quasi label *of π. If τ is the quasi label of π, then (u, τ, v) is the* quasi profile *of π.*

Every a-path π is a quasi transition. If $\text{Label}(\pi) = a^{i_1}a^{i_2}\dots a^{i_k}$, then the quasi label of π is a^j, where $j = i_1 + \dots + i_k$.

Example 3.10. If q_1, q_2 are paths

$$\langle(o, a^2, u_1), (u_2, a^{-3}, u_2), (u_3, a^5, u_3)\rangle,$$
$$\langle(u_3, b^{-7}, u_4), (u_4, b^{11}, u_5), (u_5, b^{-13}, o)\rangle,$$

respectively, then q_1 is a quasi transition from o to u_3 with quasi-label a^4, and q_2 is a quasi transition from u_3 to o with quasi-label b^{-9}. According to the paths definition in section 2.2, the labels of q_1 and q_2 are $a^2a^{-3}a^5$ and $b^{-7}b^{11}b^{-13}$, respectively.

DEFINITION 3.11 (quasi runs). *A sequence Q of quasi-transitions $q_1, \dots, q_\ell$ is a* quasi run *if the initial vertex of q_1 is o, every q_{i+1} starts at the final state of q_i, and the final vertex of q_ℓ is o. The* label *of Q is the concatenation of the quasi labels of the constituent quasi transitions. The concatenation $q_1 + \cdots + q_\ell$ of the paths $q_1, \dots, q_\ell$ is the* associate run *of Q.*

Our definition of quasi runs is narrow in the sense that the initial state o is the start and end of any quasi run. We will not need more general quasi-runs.

COROLLARY 3.12 (quasi-run labels). *Let Q be a quasi run and π be the associate run. Then π is an accepting run, and the label of π is equal, as a group element, to the label of Q.*

DEFINITION 3.13 (tolerance). *A recognizer* tolerates *an exponent word w if there is a quasi run with label w.*

Example 2.16 (continuation). The recognizer tolerates a^4b^{-9}. This is witnessed by the sequence $Q = \langle q_1, q_2 \rangle$. The concatenation $q_1 + q_2$ of the paths q_1, q_2 is the run

$$(o, a^2, u_1), (u_2, a^{-3}, u_2), (u_3, a^5, u_3), (u_3, b^{-7}, u_4), (u_4, a^{11}, u_5), (u_5, a^{-13}, o)$$

that accepts the word $a^2a^{-3}a^5b^{-7}b^{11}b^{-13}$ equal to a^4b^{-9} in G.

LEMMA 3.14 (membership criterion). *Consider a recognizer R, and let w be an exponent word. The following are equivalent:*

1. *The group element w belongs to $\Gamma(R)$.*
2. *R tolerates the reduct of w.*

Proof.

$2 \to 1$: Suppose that R tolerates w. Then w is the label of a quasi run of Q. By the quasi-run labels corollary, the associate run of Q is accepting, and its label is equal to w. Therefore $w \in \Gamma(R)$.

$1 \to 2$: Suppose that the group element w belongs to $\Gamma(R)$. Without loss of generality, $w \neq 1$. By the recognizer's subgroup definition in section 2.2, R accepts an exponent word w' equal to w. Since every accepting run is also a quasi run, R tolerates w'. Let $Q = \langle q_1, \dots, q_\ell \rangle$ be a quasi run with the fewest number of quasi transitions that tolerates an exponent word w_0 equal to w. Due to the choice of Q, every quasi label (q_i) is a syllable (rather than 1). Accordingly w_0 has the form $a_1^{p_1} \dots a_\ell^{p_\ell}$. We show that w_0 is reduced.

If w_0 is not reduced, then $a_{i+1} = a_i$ for some i. Let Q' be the quasi run obtained from Q by replacing $\langle q_i, q_{i+1} \rangle$ with a single quasi-transition $q_i + q_{i+1}$. The label of Q' is equal to w_0 but has fewer syllables, which contradicts the choice of Q. ◻

3.3. Reading algorithm.

DEFINITION 3.15 (fat). *Let R be a recognizer. For every original letter a and every vertex v,*

- $\mathrm{Fat}_R(a^+, v) = \max(0, p-1)$*, where p is the number of positive a-edges from v;*
- $\mathrm{Fat}_R(a^-, v) = \max(0, n-1)$*, where n is the number of negative a-edges from v;*
- $\mathrm{Fat}_R(a, v) = \max(0, p-1) + \max(0, n-1)$*;*
- $\mathrm{Fat}_R(v) = \sum_a \mathrm{Fat}_R(a, v)$ *and* $\mathrm{Fat}(R) = \sum_v \mathrm{Fat}(v)$.

The subscript is omitted if the context uniquely defines the recognizer.

DEFINITION 3.16 (lean recognizers). *A recognizer is* lean *if, for every original letter a and every vertex v, there is at most one positive a-edge and at most one negative a-edge coming from v.*

Obviously, lean recognizers are deterministic. However deterministic recognizers are not necessarily lean. A deterministic recognizer may have two positive a-edges coming from the same vertex if their labels are distinct syllables. Recall that inputs are syllables, not letters.

Lemma 3.17 (partisan quasi transitions).

(A) *If a lean recognizer R tolerates a reduced exponent word w, then there is a quasi-run Q with label w such that every constituent quasi transition of Q is partisan.*

(B) *Let R be a lean recognizer. For any state u and syllable a^j, there is at most one partisan quasi-transition q from u with quasi-label a^j.*

(C) *There is a polynomial-time algorithm that, given a lean recognizer R, a state u of R, and a syllable a^j, determines whether there exists a partisan quasi-transition q from u with quasi-label a^j and, if yes, finds an appropriate q.*

Proof. (A) Suppose that R tolerates a reduced exponent word $w = a_1^{p_1} \cdots a_k^{p_k}$. By the tolerance definition, there is a quasi run with label w. Let Q be a quasi-run $\langle q_1, \dots, q_k\rangle$ with label w such that the associate run of Q is as short as possible. If some q_i is not partisan, then it has successive a_i-edges e, and f of different signs. The end-vertex of e has outgoing a_i-edges e^{-1} and f. Since R is lean, $e^{-1} = f$, and therefore both edges can be removed from q_i without changing the quasi profile of q. This contradicts the choice of Q.

(B) Assume that $j > 0$; the case $j < 0$ is similar. By contradiction assume that there exist distinct partisan quasi transitions with the same quasi-label a^j from u. Then we have two distinct positive a-paths of the same length from the same vertex u. Since neither path can be a prefix of the other, there is a vertex v where the two paths branch out. Then there are two distinct positive a-edges from v, which contradicts the leanness of R.

(C) We describe the desired algorithm. Assume $j > 0$; the case $j < 0$ is similar. Let $u_0 = u$. Due to the fact that R is lean, for any nonnegative integer k, there is at most one positive a-path π_k of the form

$$\langle (u_0, a^{p_1}, u_1), (u_1, a^{p_2}, u_2), \dots, (u_{k-1}, a^{p_k}, u_k)\rangle.$$

Let k be the least number such that $\text{Length}(\pi_k) \geq j$ or else π_k does not exist. If $\text{Length}(\pi_k) = j$, then π_k is the desired quasi transition; otherwise, the desired quasi transition does not exist.

From the complexity point of view, one may worry about the case in which the vertices u_i are not distinct. In such a case, consider the very first vertex repetition on π_k. Let u_i be the first π_k vertex on the loop, $\ell = \text{Length}(\pi_k[u_0, u_i])$, and m be the length of the loop. The problem reduces to the case without vertex repetition where u_i plays the role of u_0 and $(j - \ell) \mod m$ plays the role of j. ∎

Lemma 3.18 (reading). *There is a polynomial-time* reading algorithm *that, given a lean recognizer R and an exponent word w, determines whether the group element w belongs to $\Gamma(R)$.*

Proof. Use the exponent word reduction algorithm of section 3.1 to compute the reduct $w_0 = a_1^{p_1} \dots a_k^{p_k}$ of w. By the membership criterion of section 3.2, $\Gamma(R)$ contains the group element w if and only if R tolerates w_0. We need to determine whether there is a quasi-run Q with label w_0.

By the partisan quasi-transitions lemma, part (A), we may restrict attention to quasi-runs $Q = \langle q_1, \dots, q_k\rangle$ such that every q_i is partisan. By part (B) of the same lemma, there is at most one such Q. Let A be the algorithm of part (C) of the same lemma.

The reading algorithm iterates A and has at most k rounds. Assume that the reading algorithm has performed i rounds and in the process has constructed an initial segment $\langle q_1, \ldots, q_i \rangle$ of the desired quasi-run Q. The assumption trivially holds in case $i = 0$. If $i = 0$, let $u_0 = o$; otherwise, let u_i be the end-vertex of q_i.

In case $i < k$, the reading algorithm starts round $i+1$ by applying A to R, u_i, and $a_{i+1}^{p_{i+1}}$. If A determines that there is no partisan quasi transition with a quasi label from u_i with a label of the form $a_{i+1}^{p_{i+1}}$, then the desired quasi-run Q does not exist. Otherwise let q_{i+1} be the partisan quasi transition constructed by A.

In case $i = k$, the desired quasi-run Q exists if and only $u_k = o$. $\square$

In the remaining part of this section, we construct a polynomial-time algorithm that transforms any recognizer to an equivalent lean recognizer.

3.4. Path folding. In section 2 we used edge folding to reduce a recognizer to a lean one. Now the situation is more involved, and edge folding is not going to do the job. Instead we will be folding paths.

Recall the branches, cycles, and nooses definition in section 2.2, and fix an original letter a.

DEFINITION 3.19 (impasse). *A nonempty partisan a-path is an* impasse *if it is a branch and its end-vertex has no outgoing a-edges of the sign of π.*

DEFINITION 3.20 (closed path). *A nonempty partisan a-path is* closed *if it is an impasse, a cycle, or a noose.*

LEMMA 3.21 (closed path). *Every a-edge e_1 gives rise to a closed partisan a-path $\pi = \langle e_1, \ldots, e_k \rangle$ that continues e_1. Furthermore, there is an algorithm that constructs such a path in a time polynomial in the recognizer's size.*

Proof. The desired algorithm is iterative; we describe one round of it. Suppose that we have already an a-path $\langle e_1, \ldots, e_i \rangle$ with vertex sequence $\langle u_0, \ldots, u_i \rangle$. If $u_i \in \{u_0, \ldots, u_{i-1}\}$ or if u_i has no outgoing a-edge of the same sign as e_1, then halt. Otherwise let e_{i+1} be the lexicographically first a-edge from u_i of the same sign as e_1. $\square$

As usual g.c.d.(m, n) is the greatest common divisor of positive integers m and n. Define $m = n \mod \infty$ to be equivalent to $m = n$.

DEFINITION 3.22 (two-path divisor). *For $i = 1, 2$, let π_i be a branch or cycle of length ℓ_i. We define the* divisor *as follows:*

$$\mathrm{Div}(\pi_1, \pi_2) = \begin{cases} \infty & \textit{if } \pi_1, \pi_2 \textit{ are branches}, \\ \ell_i & \textit{if } \pi_i \textit{ is a cycle and } \pi_{3-i} \textit{ is a branch}, \\ \text{g.c.d.}(\ell_1, \ell_2) & \textit{if } \pi_1, \pi_2 \textit{ are cycles.} \end{cases}$$

DEFINITION 3.23 (entanglement). *Suppose that π and ρ are nonempty partisan a-paths of the same sign and with the same initial vertex ν, and suppose that either path is a branch or a cycle. The two paths are* entangled *if there exist vertices u and v on π and ρ, respectively, such that $u \neq v$ and*

$$\mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho).$$

Otherwise the two paths are disentangled.

COROLLARY 3.24 (entanglement algorithm). *There is a polynomial-time algorithm that, given a recognizer and two paths π and ρ as in the entanglement definition, determines whether they are entangled. Furthermore, if the paths are entangled, then the algorithm produces vertices u and v witnessing the entanglement and identifies them.*

We omit the pretty obvious proof.

Lemma 3.25 (entanglement). *If π and ρ are entangled and vertices u and v witness the entanglement, then the identification of u and v does not change the recognizer subgroup.*

Proof. Let $d = \mathrm{Div}(\pi, \rho)$, $k = \mathrm{Length}(\pi[\nu, u])$, $\ell = \mathrm{Length}(\rho[\nu, v])$, and $H = \mathrm{Coset}(\nu)$. We assume that π and ρ are positive; the negative case is similar. By the coset stability lemma in section 2.4, $\mathrm{Coset}(u) = Ha^k$, and $\mathrm{Coset}(v) = Ha^\ell$. By the vertex identification lemma in section 2.4, it suffices to prove that $Ha^k = Ha^\ell$. Since u and v witness the entanglement, we have $k = \ell \mod d$.

Case 1: Both paths are branches. Then $k = \ell$, and therefore $Ha^k = Ha^\ell$.

Case 2: One of the paths is a branch, and the other is a cycle. Without loss of generality, π is a circle, and ρ is a branch. Then $Ha^k = H$, and $\ell = p \cdot k$ for some integer p. Then $Ha^\ell = H(a^k)^p = H = Ha^k$.

Case 3: Both paths are cycles. Then $Ha^k = Ha^\ell = H$. Since $d = \text{g.c.d.}(k, \ell)$, there are integers i and j such that $ik + j\ell = d$, and so $Ha^d = H(a^k)^i(a^\ell)^j = H$. Since $k = \ell \mod d$, there is an integer p such that $k = pd + \ell$. Then $Ha^k = H(a^d)^p a^\ell = Ha^\ell$. ∎

Recall the disjoint paths definition in section 2.2.

Definition 3.26 (path folding). *Suppose that π and ρ are nonempty partisan a-paths such that*

- *they have the same sign and the same start-vertex ν,*
- *either path is a branch or a cycle,*
- $\mathrm{Length}(\pi) \geq \mathrm{Length}(\rho)$ *if both paths are branches, and*
- $\mathrm{Length}(\pi) = \mathrm{Div}(\pi, \rho)$ *if both paths are cycles.*

To fold ρ into π, *execute the following* folding algorithm:

1. *Apply the entanglement algorithm to π and ρ. If the number of vertices is reduced, then halt. Otherwise the paths are disentangled.*
2. *For each v on ρ, create a new vertex v' on π such that*

$$\mathrm{Length}(\pi[\nu, v']) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho)$$

unless π has a vertex at this position already.

3. *Identify every clone v' with its original v.*
4. *Remove all edges of ρ and their inverses.*

One of our referees suggested "path merging" instead of "path folding."

Remark 3.27 (path folding). Concerning stage 2 of the folding algorithm, consider the case in which π has a vertex u such that $\mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho)$. Since π and ρ are disentangled (otherwise we would not arrive at stage 2), we have $u = v$. Assume that π and ρ are internally disjoint. Then vertex v is an extreme vertex on both paths. Consider the scenario in which v differs from the initial vertex ν. Then v is the final vertex of both π and ρ, both π and ρ are branches, and $\mathrm{Length}(\pi) = \mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) = \mathrm{Length}(\rho)$.

Lemma 3.28 (path folding). *Let π and ρ be as in the path folding definition. Folding ρ into π preserves the recognizer subgroup. If the algorithm halts at stage* 1*, then the number of vertices of the recognizer decreases. Otherwise the number of vertices is unchanged, and the (amount of) fat changes as follows.*

(BB) *Suppose that π and ρ are branches of lengths m and n, respectively.*
If $m = n$, then the fat decreases by 2.
If $m > n$ and ρ is an impasse, then the fat decreases by 1.
If $m > n$ but ρ is not an impasse, then the fat does not change.

(CB) *Suppose that π is a cycle and ρ is a branch.*
If ρ is an impasse, then the fat decreases by 1*;*
otherwise, the fat does not change.

(CC) *Suppose that π and ρ are cycles. Then the fat decreases by* 2.

Proof. We consider only the case in which π and ρ are positive; the case in which they are negative is similar.

Let R be the given recognizer, and for $p \leq 4$, let R_p be the recognizer obtained from R by executing p stages of the folding algorithm, so that $R_0 = R$. Let the vertex sequence of π be $\langle u_0, \ldots, u_k\rangle$. Let $\rho = \langle f_1, \ldots, f_\ell\rangle$ and the vertex sequence of ρ be $\langle v_0, \ldots, v_\ell\rangle$. Let $d = \mathrm{Div}(\pi, \rho)$. The case in which π and ρ are entangled is obvious. Assume that π and ρ are disentangled. Then nothing happens at stage 1, and so $R_1 = R_0$.

First we note that the vertices of R_4 are those of R_1. Indeed all vertices v'_j created at stage 2 are identified with the respective vertices v_j at stage 3; thus, the vertices of R_3 are those of R_1. And the vertices do not change at stage 4.

Second we show that $\Gamma(R_4) = \Gamma(R)$. By the vertex creation lemma, $\Gamma(R_2) = \Gamma(R_1)$. By the vertex identification lemma of section 2.4, $\Gamma(R_3) = \Gamma(R_2)$. It remains to show that $\Gamma(R_4) = \Gamma(R_3)$. Let $n_j = \mathrm{Length}(\pi[v_0, v_j])$, $r_j = n_j \mod d$, and $v'_0 = u_0$. It suffices to show that for every edge f_j of ρ, R_4 has a path P_j with the profile of f_j. Indeed, suppose we have the desired paths P_j. Recall that the profile of a path includes not only the label but also the initial and final vertices, so the desired paths P_j match up appropriately to simulate ρ. It is easy to see that, for every vertex v, $\mathrm{Coset}(v)$ computed in R_3 is the same as the one computed in R_4.

The profile of f_j is (v_{j-1}, a^p, v_j), where $p = n_j - n_{j-1}$. In scenario (BB), the desired path P_j is $\pi[v'_{j-1}, v'_j]$. In scenarios (CB) and (CC), $p = d \cdot q + (r_{j+1} - r_j)$ for some q. The desired path P_j starts at v'_{j-1} and ends at v'_j. If $r_j \leq r_{j+1}$, then π does q full revolutions around π. If $r_j > r_{j+1}$, then $q > 0$, and π does $q-1$ full revolutions around π.

Finally we prove the claims about the fat. By the vertex creation lemma, $\mathrm{Fat}(R_2) = \mathrm{Fat}(R_1)$. Thus we need to examine only the evolution of the fat from R_2 to R_4. Furthermore, it suffices to examine the evolution of the numbers $\mathrm{Fat}(a, v_j)$. Indeed, this will account for the vertices v'_j identified with the corresponding vertices v_j at stage 3. If a vertex v of R_2 differs from any v_j, v'_j, then the immediate vicinity of v does not change. If an original letter $b \neq a$, then $\mathrm{Fat}(b, v_j)$ does not change. The reason is that, upon the creation, the vertex v'_j has no b-edges adjacent to it. As a result the identification of v_j with v'_j creates no b-fat.

If $0 < j < \ell$, then $\mathrm{Fat}(a, v_j)$ does not change from R_2 to R_4. Indeed, as a result of identification with v'_j, the vertex v_j acquires one outgoing positive a-edge and one outgoing negative a-edge at stage 3, but then, at stage 4, it loses one outgoing positive a-edge, namely, f_{j+1}, and one outgoing negative a-edge, namely, f_j^{-1}.

The vertex v_0 does not acquire any outgoing edges at stage 3, and thus neither $\mathrm{Fat}(a^+, v_0)$ nor $\mathrm{Fat}(a^-, v_0)$ increase at stage 3. It loses one positive outgoing edge, namely, f_0 at stage 4, and so $\mathrm{Fat}(a^+, v_0)$ decreases by 1 from R_2 to R_4. It does not lose any negative outgoing edge in scenarios (BB) or (CB), and so $\mathrm{Fat}(a^-, v_0)$ does not change in scenarios (BB) and (CB). Since $v_0 = v_\ell$ in scenario (CC), it remains to examine only the evolution of the numbers $\mathrm{Fat}(a^+, v_\ell)$ and $\mathrm{Fat}(a^-, v_\ell)$ in the three scenarios.

(BB) In this scenario, we first suppose that $m = n$. Since π and ρ are disentangled, $u_k = v_\ell$. The vertex v_ℓ does not acquire any outgoing edges at stage 3 and loses only one outgoing edge, namely, the negative edge f_l^{-1}, at stage 4. Thus $\mathrm{Fat}(a^+, v_\ell)$

does not change, and $\mathrm{Fat}(a^-, v_\ell)$ decreases by 1. To summarize, $\mathrm{Fat}(a^+, v_0)$ and $\mathrm{Fat}(a^-, v_\ell)$ decrease by 1 while $\mathrm{Fat}(a^-, v_0)$ and $\mathrm{Fat}(a^+, v_\ell)$ do not change. Hence $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$.

Second we suppose that $m > n$. At stage 3, v_ℓ acquires one positive and one negative outgoing edge. At stage 4, v_ℓ loses no outgoing positive edges but loses f_ℓ^{-1}. Thus $\mathrm{Fat}(a^-, v_\ell)$ does not change. If ρ is not an impasse, then $\mathrm{Fat}(a^+, v_\ell)$ increases by 1; otherwise, $\mathrm{Fat}(a^+, v_\ell)$ remains zero throughout the process. We summarize. If ρ is an impasse, then $\mathrm{Fat}(a^+, v_0)$ decreases by 1 while $\mathrm{Fat}(a^-, v_0)$, $\mathrm{Fat}(a^+, v_\ell)$, and $\mathrm{Fat}(a^-, v_\ell)$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 1$. If ρ is not an impasse, then $\mathrm{Fat}(a^+, v_0)$ decreases by 1, $\mathrm{Fat}(a^+, v_\ell)$ increases by 1, and $\mathrm{Fat}(a^-, v_0)$ and $\mathrm{Fat}(a^-, v_\ell)$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2)$.

(CB) This scenario is similar to the case $m > n$ of scenario (BB). Let us just point out that the vertex v_ℓ does not occur on π in R. Indeed suppose the opposite. Since π and ρ are internally disjoint and $u_k = u_0$, we have $v_\ell = u_0 = v_0$. But then ρ is a cycle which contradicts scenario (CB).

(CC) In this scenario, $v_0 = v_\ell$, and thus $\mathrm{Fat}(a^+, v_\ell)$ decreases by 1. $\mathrm{Fat}(a^-, v_\ell)$ does not change at stage 3 and decreases by 1 at stage 4 because f_ℓ^{-1} is removed. To summarize, the overall change in the fat is this: both $\mathrm{Fat}(a^+, v_\ell)$ and $\mathrm{Fat}(a^-, v_\ell)$ decrease by 1. Thus $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$. □

3.5. Weight reduction algorithm.

DEFINITION 3.29 (recognizer weight). *The* weight *of a recognizer R is a pair (i, j) of natural numbers where i is the number of vertices of R and $j = \mathrm{Fat}(R)$. The weights are ordered lexicographically with the number of vertices being the more significant component.*

LEMMA 3.30 (weight reduction). *There is a polynomial-time* weight reduction algorithm *that reduces any recognizer to an equivalent lean recognizer.*

Proof. We construct an iterative algorithm that transforms the given recognizer by means of path folding; the algorithm halts when the recognizer is lean. By the folding lemma, the algorithm preserves the recognizer subgroup.

We describe one round of the algorithm and show that the weight decreases at each round. It will be obvious that the algorithm is polynomial time.

If the current recognizer R is lean, halt. Otherwise find a quadruple (a, ξ, e_1, f_1), where a is an original letter, ξ is a vertex with $\mathrm{Fat}(a, \xi) > 0$, and e_1 and f_1 are two a-edges from ξ of the same sign. It could be the lexicographically first such quadruple, for example. We consider only the case in which e_1 and f_1 are positive; the case in which they are negative is similar. Use the algorithm of the closed paths lemma to construct closed a-paths E and F continuing the e_1 and e_2, respectively. We consider first the cases in which E and F are disjoint off ξ and then the other cases.

Part 1: Assume that E and F are disjoint off ξ.

Case 1: E and F are cycles.

If the cycles are of different lengths, let π be the shorter one; otherwise, let π be either of the cycles. Let ρ be the other cycle, $m = \mathrm{Length}(\pi)$, and $n = \mathrm{Length}(\rho)$. If m divides n, fold ρ into π. Otherwise, let d be the greatest common divisor of m and n. Create a positive single-edge a-cycle λ of length d at ξ. Then fold π into λ, and let π' be the resulting cycle of length d. Then fold ρ into π'.

To examine the evolution of weight, we apply scenario (CC) of the folding lemma to the following two subcases.

First suppose that m divides n. If π and ρ are entangled, then the number of vertices drops. Otherwise the number of vertices is unchanged, but the fat decreases by 2.

Second suppose that m does not divide n. If π' and ρ are entangled, then the number of vertices drops. Suppose that π' and ρ are disentangled. With the creation of λ, the vertex ξ acquires one outgoing positive a-edge and one outgoing negative a-edge, so that the fat increases by 2. The folding of π into λ decreases the fat by 2. The folding of ρ into π' decreases the fat by 2. Altogether the fat decreases by 2.

Case 2: One of the paths E and F is a cycle, and the other is an impasse.

Fold the impasse into the cycle. By scenario (CB) of the folding lemma, this decreases the recognizer weight.

Case 3: One of the paths E and F is a cycle, and the other is a noose.

Let π be the cycle, ρ be the noose, and v be the end-vertex of ρ. Fold the tail of ρ into π, and let π' be the resulting cycle. Note that v occurs on π'. By scenario (CB) of the folding lemma, this does not increase the recognizer weight. If the weight dropped, then finish the round. Otherwise let π'' be the cycle at v obtained from π' by redefining the initial vertex as v. The loop of ρ is another cycle at v. Clearly the two cycles at v are disjoint off ξ. Proceed as in case 1.

Case 4: E and F are impasses.

If the impasses are of different length, let π be the longer one; otherwise, let π be either of the impasses. Let ρ be the other impasse. Fold ρ into π. By scenario (BB) of the folding lemma, this decreases the recognizer weight.

Case 5: One of the paths E and F is an impasse, and the other is a noose.

Let π be the impasse, ρ the noose, τ the tail of ρ, λ the loop of ρ, $m = \text{Length}(\pi)$, and $n = \text{Length}(\tau)$. If $m \leq n$, then fold π into τ; by scenario (BB) of the folding lemma, this decreases the weight. Suppose that $m > n$. Then fold τ into π, and let π' be the resulting impasse. By the folding and weight corollary, this does not increase the weight. If the weight decreases, finish the round. Otherwise let v be the final vertex of τ and π'' be the suffix π' with initial vertex v. Obviously, λ is a cycle at v, π'' is an impasse with initial vertex v, and the two paths are disjoint off v. Proceed as in case 2.

Case 6: E and F are nooses.

Let π be the noose with a shorter tail. If the two tails are of the same length, let π be either noose. Let ρ be the other noose and u be the final vertex of π. Fold the tail of π into the tail of ρ, and let R' be the resulting recognizer. By the folding and weight corollary, $\text{Weight}(R') \leq \text{Weight}(R)$. If $\text{Weight}(R') < \text{Weight}(R)$, finish the round. Otherwise let π' be the loop of π and ρ' be the suffix of ρ with initial vertex u. π' is a cycle at u, ρ' is either a cycle at u or a noose with initial vertex u, and the two paths are disjoint. If ρ' is a cycle, proceed as in case 1. If ρ' is a noose, proceed as in case 3.

Part 2: Assume that E and F are not disjoint off ξ.

Case 7: At least one of the paths E and F is a cycle.

Let π be the cycle or one of the two cycles, and let ρ be the maximal initial segment of the other path that is internally disjoint from π. The final vertex v of ρ splits π into two segments $\pi_1 = \pi[\xi, v]$ and $\pi_2 = \pi[v, \xi]$. Without loss of generality, we may assume $\text{Length}(\pi_1) \geq \text{Length}(\rho)$; otherwise, swap π_1 and ρ in the remainder of this case. Let $m = \text{Length}(\pi_1)$ and $n = \text{Length}(\rho)$.

Fold ρ into π_1, and let R' be the resulting recognizer. If the weight decreases, then end the current round of the algorithm. Suppose that the weight does not decrease. Then, by the folding lemma, $m > n$. We have two internally disjoint cycles at v in R'. One is formed by the suffix of length $m - n$ of π_1. The other is formed by the concatenation of π_2 and the prefix of length n of π_1. Proceed as in case case 1.

Case 8: At least one of the paths E and F is a noose.

Let τ be a noose or one of the two nooses, τ_1 and τ_2 be the tail and loop of τ, respectively, and v be the final vertex of τ_1. Recall that every path P gives rise to a reverse path P^{-1}. Let π be the cycle τ_2^{-1} at v. By an argument similar to the proof of the closed path lemma, there is a closed path ρ that continues the path τ_1^{-1}. Thus we have two closed paths, π and ρ, sharing the same initial vertex v and having the same sign (that is both positive or both negative). If π and ρ are internally disjoint, proceed according to the appropriate case of part 1. Otherwise proceed as in case 7.

Case 9: Both E and F are impasses.

Let F_1 be the maximal initial segment of F disjoint from E. The final vertex v of F_1 splits E into the prefix $E_1 = E[\xi, v]$ and the corresponding suffix E_2. If $\text{Length}(E_1) \geq \text{Length}(F_1)$, let $\pi = E_1$ and $\rho = F_1$; otherwise, let $\pi = F_1$ and $\rho = E_1$. Let $m = \text{Length}(\pi)$ and $n = \text{Length}(\rho)$. Fold ρ into π. If the recognizer weight decreases, then finish the round of the algorithm. Suppose that the weight does not decrease. Then, by the folding lemma, $m > n$. We have two internally disjoint closed paths of the same sign with initial vertex v. One is the cycle formed by a suffix of length $m - n$ of π. The other is the impasse E_2. Proceed as in case 2. This completes the proof of the lemma. ☐

3.6. The theorem. We restate the free groups with elements in exponent normal form theorem formulated in section 1.

Theorem 3.31. *There is a polynomial-time decision algorithm for the membership problem for the succinct free group. More explicitly, there is an algorithm such that*

(i) *given exponent words $h_1, \ldots, h_m$ and w, the algorithm decides whether the subgroup H generated by $h_1, \ldots, h_m$ contains w, and*
(ii) *the algorithm runs in time polynomial in $|h_1| + \cdots + |h_m| + |w|$.*

Proof. Use the construction algorithm of section 2.2 to produce a recognizer R_1 for H. Then use the weight reduction algorithm of section 3.5 to transform R_1 into a lean recognizer R_2. Finally use the reading algorithm of section 3.3 to check whether the group element w belongs to H. Since all the constituent algorithms are polynomial time, the decision algorithm is polynomial time. ☐

4. The free product of $(\mathbb{Z}/2\mathbb{Z})$ and $(\mathbb{Z}/3\mathbb{Z})$. This auxiliary section aims to clarify certain aspects of the membership problem for the modular group unrelated to the matrix representation of the modular group.

In the notation of combinatorial group theory, $\langle g \mid g^n \rangle$ is a cyclic group of order n with generator g. It is isomorphic to the additive group $\mathbb{Z}/n\mathbb{Z}$ of integers modulo n. The modular group is isomorphic to the free product $(\mathbb{Z}/2\mathbb{Z}) * (\mathbb{Z}/3\mathbb{Z})$; see [8] for a clear explanation of this fact. We use the syllabic recognizer approach to give a polynomial time decision procedure for the membership problem for the group $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ in the following presentation.

Definition 4.1 (standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$). Standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ *is the free group in the following syllabic presentation.*

- *Alphabet: The alphabet consists of original letters s, t and one auxiliary symbol ${}^{-1}$. The alphabet is linearly ordered in some way; it will play no role in what order it is exactly.*
- *Syllables:* Syllables *are s, t, and t^{-1}. The Greek letter ε is reserved to range over $\{1, -1\}$, so that t^ε is always either t or t^{-1}.*
- *Inverses: Syllables t and t^{-1} are the inverses of each other, and s is its own inverse.*

- *Equality: It is the least congruence for the word semigroup with the inverse operation such that $ss = 1, tt^{-1} = 1, t^{-1}t = 1, tt = t^{-1}$, and $t^{-1}t^{-1} = t$.*

It is easy to see that the quotient of the word semigroup is indeed isomorphic to $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. Consider the five equalities that appear in the definition of the equality relation. Reading each of the five equalities from left to right gives us five different reduction steps. It is easy to see that any two words w_1, w_2 are equal if and only if there is a *witness sequence* of words $x_0, x_1, \dots, x_\ell$ such that $x_0 = w_1$, $x_\ell = w_2$ and, for every two successive words x_i, x_{i+1}, one is obtained from the other by a single reduction step.

A word w is *reduced* if it does not contain any of the "forbidden" contiguous substrings $ss, tt^{-1}, t^{-1}t, tt$, or $t^{-1}t^{-1}$. The group G in consideration is the quotient of W with respect to the equality relation.

LEMMA 4.2 (word reduction).

1. *For every word w, there is a unique reduced word equal to w. The reduced word is the* reduct *of w.*
2. *There is a polynomial-time* word reduction algorithm *that, given a word w, produces the reduct of w.*

We omit the proof of this well-known fact.

DEFINITION 4.3 (irregular paths). *There are two kinds of irregular paths.*

- *An* s-irregular path *has the form $\langle (u_0, s, u_1), (u_1, s, u_2)\rangle$, where $u_0 \neq u_2$.*
- *An* t-irregular path *has the form $\langle (u_0, t, u_1), (u_1, t, u_2), (u_2, t, u_3)\rangle$, where $u_0 \neq u_3$.*

DEFINITION 4.4 (regular recognizers). *A recognizer without irregular paths is* regular.

LEMMA 4.5 (irregular paths). *The identification of the end-vertices of an irregular path produces an equivalent recognizer.*

Proof. Due to the vertex identification lemma of section 2.4, it suffices to show that the end-vertices of the given irregular path π have the same associated coset. To this end we use the coset stability lemma of section 2.4. If π has the form $\langle (u, s, u_1), (u_1, s, v)\rangle$, then $\mathrm{Coset}(v) = \mathrm{Coset}(u)ss = \mathrm{Coset}(u)$. And if π has the form $\langle (u, t, u_1), (u_1, t, u_2), (u_2, t, v)\rangle$, then $\mathrm{Coset}(v) = \mathrm{Coset}(u)ttt = \mathrm{Coset}(u)$. ◻

DEFINITION 4.6 (fat). *Let R be a regular recognizer and u be a state of R. For every syllable σ, $\mathrm{Fat}_R(\sigma, u) = \max(0, n-1)$, where n is the number of σ-labeled transitions from u. The subscript may be omitted when the recognizer is uniquely determined by the context.*

A recognizer R is deterministic if every $\mathrm{Fat}_R(\sigma, u) = 0$. Note that a deterministic recognizer does not have s-irregular paths.

DEFINITION 4.7 (triangles). *A* triangle *in a recognizer is a cycle of the form $\langle (u_1, t, u_2), (u_2, t, u_3), (u_3, t, u_1)\rangle$. We allow the possible "degenerate" cases where some of the edges coincide. In particular, if $e = (v, t, v)$, then $\langle e, e, e\rangle$ is a triangle. An* incomplete triangle *is a two-edge path $\langle (u_1, t, u_2), (u_2, t, u_3)\rangle$ such that there is no edge (u_3, t, u_1).*

DEFINITION 4.8 (triangle complete). *A recognizer is* triangle complete *if it does not have any incomplete triangles.*

Note that if a recognizer is triangle complete, then for any edges $e_1 = (u_1, t^{-1}, u_2)$ and $e_2 = (u_2, t^{-1}, u_3)$, there is an edge $e_3 = (u_3, t^{-1}, u_1)$. Indeed, by the triangle completeness requirement, applied to $\langle e_2^{-1}, e_1^{-1}\rangle$, there is an edge (u_1, t, u_3). But its inverse is the desired (u_3, t^{-1}, u_1).

LEMMA 4.9 (completing one triangle). *Suppose that a deterministic regular recognizer R has an incomplete triangle $\langle (u_1, t, u_2), (u_2, t, u_3)\rangle$. Add a new edge (u_3, t, u_1) and its inverse, and let S be the resulting recognizer. Then*

1. *S is equivalent to R;*
2. *S is deterministic;*
3. *S is regular.*

Proof. Let $e_1 = (u_1, t, u_2)$, $e_2 = (u_2, t, u_3)$, and $e_3 = (u_3, t, u_1)$.

1. Obviously $\Gamma(R) \subseteq \Gamma(S)$. We show that $\Gamma(S) \subseteq \Gamma(R)$. If π is a run of S that accepts a word w, replace every occurrence of e_3 (respectively, e_3^{-1}) in π with $\langle e_2^{-1}, e_1^{-1}\rangle$ (respectively, $\langle e_1, e_2\rangle$). The result is a run of R that accepts a word equal to w in G.

2. By contradiction assume that S is not deterministic. Taking into account that R is deterministic and that e_3 and e_3^{-1} are the only new edges of S, we have $\mathrm{Fat}_S(t^{-1}, u_1) > 0$ or $\mathrm{Fat}_S(t, u_3) > 0$. Either case leads to a contradiction. We consider only the first case; the second case is similar.

Suppose that $\mathrm{Fat}_S(t^{-1}, u_1) > 0$. Then vertex u_1 has an outgoing edge of the form (u_1, t^{-1}, u_0) in R. Let $e_0 = (u_0, t, u_1)$. Since R is regular, the end-vertices of the path $\langle e_0, e_1, e_2\rangle$ coincide, so that $u_0 = u_3$ and $e_0 = (u_3, t, u_1)$, which is impossible because $\langle e_1, e_2\rangle$ is an incomplete triangle in R.

3. If S is not regular, then it has an irregular path π. We have proved already that S is deterministic. It follows that π cannot be s-irregular. So π is t-irregular. Let $\pi = \langle f_1, f_2, f_3\rangle$. Since R is regular, π contains the new edge e_3. We have three different scenarios $e_3 = f_i$. Each of the scenarios leads to a contradiction.

We consider here only the scenario $e_3 = f_2$; the other scenarios are similar. Taking into account that S is deterministic, we have that $f_1 = e_2$ and $f_3 = e_1$. But then u_2 is the initial and final vertex of π, which contradicts the irregularity of π. ◻

COROLLARY 4.10 (triangle completion). *There is a polynomial-time* triangle completion algorithm *that transforms an arbitrary deterministic regular recognizer into an equivalent recognizer that is regular, deterministic, and triangle complete.*

Proof. If there is an incomplete triangle $\langle (u_1, t, u_2), (u_2, t, u_3)\rangle$, then add a new edge (u_3, t, u_1) and its inverse. Keep doing that until the recognizer is triangle complete. ◻

LEMMA 4.11 (membership criterion). *Let R be a triangle complete, deterministic, regular recognizer, and let w be an arbitrary word. The subgroup $\Gamma(R)$ contains the group element w if and only if R accepts the reduct of w.*

Proof. If R accepts the reduct of w, then $w \in \Gamma(R)$ by the definition of recognizers. Suppose that $w \in \Gamma(R)$, and let w_0 be a word with the fewest number of syllables accepted by R. We show that w_0 is reduced.

Let π be a run that accepts w_0. If π has the form

$$\pi_1 + \langle (v_1, \sigma, v_2)(v_2, \sigma^{-1}, v_3)\rangle + \pi_2,$$

then the middle segment can be removed from π. And if π has the form

$$\pi_1 + \langle (v_1, t^{\varepsilon}, v_2)(v_2, t^{\varepsilon}, v_3)\rangle + \pi_2,$$

then the middle segment can be replaced by the edge $(v_1, t^{-\varepsilon}, v_3)$. In either case, the resulting run accepts a word that is equal to w and that has fewer syllables than w_0, which contradicts the choice of w_0. ◻

LEMMA 4.12 (reading). *There is a polynomial-time* reading algorithm *that, given a deterministic regular recognizer R and a word w, determines whether $w \in \Gamma(R)$.*

Proof. Let R_1 be the given recognizer. Use the triangle completion algorithm to transform R_1 into an equivalent recognizer R_2 that is regular, deterministic, and triangle complete. Use the word reduction algorithm to transform w to an equal reduced word w_0. By the membership criterion lemma, $w \in \Gamma(R_2)$ if and only if R_2 accepts w_0. To determine whether R_2 accepts w_0, run R_2 on w_0. □

LEMMA 4.13 (fat reduction). *There is a polynomial-time* fat reduction algorithm *that transforms any recognizer into an equivalent deterministic regular recognizer.*

Proof. The desired algorithm is iterative. At every round it decreases the number of vertices. We describe one round of the algorithm.

1. If there an irregular path, then identify the end-vertices of the path, and start a new round.
2. If there are distinct edges with the same initial vertex and the same label, then identify their final vertices, and start a new round.
3. Halt.

Obviously the algorithm halts, and when it does the recognizer is deterministic and regular. By the vertex identification and edge folding lemmas in section 2.4, the algorithm does not change the recognizer subgroup. □

THEOREM 4.14. *There is a polynomial-time decision algorithm for the membership problem for* $(\mathbb{Z}/2\mathbb{Z}) * (\mathbb{Z}/3\mathbb{Z})$.

Proof. Use the construction algorithm of section 2.2 to produce a recognizer R_1 for H. Use the weight reduction algorithm to transform R_1 to an equivalent recognizer that is regular and deterministic. Finally use the reading algorithm to check whether $w \in \Gamma(S)$. □

5. Succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$.

In the previous section, we proved that the membership problem for standard $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ is polynomial-time decidable. Here we introduce an exponentially more succinct syllabic representation of the elements of $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ and show that the membership problem for G remains polynomial-time decidable.

5.1. Succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$: Definition and word reduction.

DEFINITION 5.1 (succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$). Succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ *is the group* $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ *in the following syllabic representation.*

- *Alphabet: The alphabet consists of original letters s and t and the following auxiliary symbols: $^{-1}$, left and right parentheses, and ^{0}, ^{1}, ..., ^{9}. The alphabet is linearly ordered in some way; it will play no role in what order it is exactly.*
- *Syllables: The syllables split into three categories.*
  - *–Positive: strings $(ts)^k t$, where k is a natural number in decimal notation,*
  - *–Negative: strings $(t^{-1}s)^k t^{-1}$, where k is a natural number in decimal notation,*
  - *–Neutral: the string s.*

  Positive and negative syllables are partisan. *The Greek letter ε is reserved to range over $\{1,-1\}$, so that t^ε is always either t or t^{-1}. To distinguish this representation of $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ from the standard representation, new words will be called* exponent words, *and the old words will be called* unary words. *Any exponent word w expands in the obvious way to a unary word called the* unary expansion *of w. For example, $(t,s)^4 s(ts)^4$ expands to $tststststststststst$.*
- *Inverses: Syllables $(ts)^k t$ and $(t^{-1}s)^k t^{-1}$ are the inverses of each other, and s is its own inverse.*

- *Equality: Exponent words are* equal *if their unary expansions are equal in the sense of the standard $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$.*

Obviously the quotient of the word semigroup over the equality relation is isomorphic to $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$.

As usual, the sign of an integer i is $1, 0$, or -1 if $i > 0$, $i = 0$, or $i < 0$, respectively, and $|i|$ is the absolute value of i.

Definition 5.2 (T notation).

$$T_i = \begin{cases} (ts)^{i-1}t & \textit{if } i > 0, \\ (t^{-1}s)^{|i|-1}t^{-1} & \textit{if } i < 0, \\ 1 & \textit{if } i = 0. \end{cases}$$

Lemma 5.3 (syllables). *Let i, j be nonzero integers.*

1. $T_i s T_j = T_{i+j}$ *if* $\operatorname{sign}(i) = \operatorname{sign}(j)$.
2.
$$T_i T_j = \begin{cases} T_{i-1}s^\alpha T_{-1} s^\beta T_{j-1} & \textit{if } \operatorname{sign}(i) = \operatorname{sign}(j) = 1, \\ T_{i+1}s^\alpha T_1 s^\beta T_{j+1} & \textit{if } \operatorname{sign}(i) = \operatorname{sign}(j) = -1, \end{cases}$$
$$\textit{where } \alpha = \begin{cases} 1 & \textit{if } \ |i| \neq 1, \\ 0 & \textit{if } \ |i| = 1, \end{cases} \quad \textit{and} \quad \beta = \begin{cases} 1 & \textit{if } \ |j| \neq 1, \\ 0 & \textit{if } \ |j| = 1. \end{cases}$$
3.
$$T_i T_j = \begin{cases} sT_{i+j} & \textit{if } \operatorname{sign}(i) = -\operatorname{sign}(j) \textit{ and } |i| < |j|, \\ 1 & \textit{if } \operatorname{sign}(i) = -\operatorname{sign}(j) \textit{ and } |i| = |j|, \\ T_{i+j}s & \textit{if } \operatorname{sign}(i) = -\operatorname{sign}(j) \textit{ and } |i| > |j|. \end{cases}$$

The proof is straightforward. We give here only a few examples.

$$\begin{aligned}
&T_2 s T_3 = (tst)s(tstst) = (ts)^4 t = T_5, \\
&T_{-2} s T_{-3} = (t^{-1}st^{-1})s(t^{-1}st^{-1}st^{-1}) = (t^{-1}s)^4 t^{-1} = T_{-5}, \\
&T_2 T_3 = (tst)(tstst) = (t)s(tt)s(tst) = T_1 s T_{-1} s T_2, \\
&T_1 T_3 = (t)(tstst) = (tt)s(tst) = T_{-1} s T_2, \\
&T_2 T_1 = (tst)(t) = (t)s(tt) = T_1 s T_{-1}, \\
&T_{-2} T_{-2} = (t^{-1}st^{-1})(t^{-1}st^{-1}) = (t^{-1})s(t^{-1}t^{-1})s(t^{-1}) = T_{-1} s T_1 s T_{-1}, \\
&T_2 T_{-3} = (tst)(t^{-1}st^{-1}st^{-1}) = st^{-1} = sT_{-1}, \\
&T_3 T_{-2} = (tstst)(t^{-1}st^{-1}) = ts = T_1 s, \\
&T_2 t_{-2} = (tst)(t^{-1}st^{-1}) = 1.
\end{aligned}$$

Definition 5.4 (reduced exponent words). *An exponent word is* reduced *if*

(R1) *every nonfinal neutral syllable is followed by a partisan syllable,*
(R2) *every nonfinal partisan syllable is followed by a neutral syllable,*
(R3) *any two partisan syllables separated only by a neutral syllable have different signs.*

For example, the exponent word $T_3 s T_{-7} s T_2$ is reduced.

Lemma 5.5 (exponent word reduction).

1. *Every exponent word w is equal in G to a unique reduced exponent word. The reduced exponent word is the* reduct *of w.*

2. *There is a polynomial-time* exponent word reduction algorithm *that, given any exponent word w, computes the reduct of w.*

Proof. It is easy to see that (i) if an exponent word is reduced, then its unary expansion is reduced in the sense of section 4 and (ii) if two reduced exponent words are equal, then the reduced unary expansions are equal. By the word reduction lemma of section 4, equal reduced unary words are identical. It follows that equal reduced exponent words have identical unary expansions. It is easy to see that equal reduced exponent words are also identical as exponent words; that is, they have the same syllables in the same order. This establishes the uniqueness part of the first claim.

In the remainder of the proof, we construct polynomial-time algorithms A_1, A_2, and A_3 transforming any exponent word w to an equal exponent word in such a way that $A_1(w)$ satisfies requirement (R1), $A_2(A_1(w))$ satisfies requirements (R1) and (R2), and $A_3(A_2(A_1(w)))$ satisfies requirements (R1), (R2), and (R3) and thus is reduced.

Taking into account that $s^2 = 1$, A_1 is obvious. Taking into account part 1 of the syllables lemma, A_3 is obvious. It remains to construct A_2.

We say that two successive syllables of an exponent word w *collide* if both syllables are partisan. Define $\mathrm{rank}(w) = (c, b)$, where c is the number of collisions in w and b is the number of partisan syllables in w. Order ranks lexicographically. A_2 is an iterative rank-reducing algorithm. It is presumed that the input exponent word satisfies A_1. In the next paragraph, we describe one round of A_2.

If w satisfies (R2) then halt. Otherwise w has the form xT_mT_ny. If $\mathrm{sign}(m) = \mathrm{sign}(n)$, use part 2 of the syllables lemma to reduce the number of collisions. If $\mathrm{sign}(m) = -\mathrm{sign}(n)$, use part 3 of the syllables lemma to reduce the rank. If (R1) is violated in the process, then apply A_1. $\square$

5.2. Deficit reduction. Recognizers were introduced in section 2.2.

DEFINITION 5.6 (edges). *An edge of a recognizer is* partisan *or* neutral *if its label is so. A partisan edge is* positive *if or* negative *if its label is so. The* length *of an edge e with label σ is the number of symbols in the unary expansion of σ. Thus* $\mathrm{Length}(e) = 2|i| - 1$ *if $\sigma = T_i$, and* $\mathrm{Length}(e) = 1$ *if $\sigma = s$.*

DEFINITION 5.7 (paths). *The* length *of a path π is the sum of the lengths of its edges. A path π is* positive *if*

- *it has at least one positive edge and no negative edges, and*
- *the positive and neutral edges of π alternate.*

Negative paths are defined similarly. Positive and negative paths are partisan.

DEFINITION 5.8 (fat). *Let R be a recognizer and u range over the vertices of R.*

- $\mathrm{Fat}_R(s, u) = \max(0, n-1)$*, where n is the number of neutral edges from u.*
- $\mathrm{Fat}_R(t, u) = \max(0, n-1)$*, where n is the number of positive edges from u.*
- $\mathrm{Fat}_R(t^{-1}, u) = \max(0, n-1)$*, where n is the number of negative edges from u.*

Further, $\mathrm{Fat}_R(u) = \mathrm{Fat}_R(s, u) + \mathrm{Fat}_R(t, u) + \mathrm{Fat}_R(t^{-1}, u)$*, and* $\mathrm{Fat}(R) = \sum_u \mathrm{Fat}_R(u)$*. The subscript may be omitted if the context uniquely defines the recognizer.*

DEFINITION 5.9 (lean recognizers). *It is* lean *if* $\mathrm{Fat}(R) = 0$.

As in section 3.3, lean recognizers are deterministic, but deterministic recognizers are not necessarily lean. Recall the regular recognizers and the triangles definitions in section 4.

DEFINITION 5.10 (deficit). *Let R be a recognizer. A vertex u_2 of R is* deficient *if there are positive edges (u_1, T_m, u_2) and (u_2, T_n, u_3) such that either $m + n > 2$ or* *else* $m = n = 1$ *and* $\langle(u_1, T_m, u_2), (u_2, T_n, u_3)\rangle$ *is an incomplete triangle. The number of deficient vertices is the* deficit $\Delta(R)$ *of* R.

Lemma 5.11 (deficit decrement). *There is a polynomial-time* deficit decrement algorithm *that transforms any lean regular recognizer* R *with positive deficit to an equivalent lean regular recognizer* S *with lesser deficit.*

Proof. Find a deficient vertex u_2 together with edges $e_1 = (u_1, T_m, u_2)$ and $e_2 = (u_2, T_n, u_3)$ witnessing the deficiency of u_2. Without loss of generality $m \geq n$; the case $n \geq m$ is similar. We consider various scenarios that arise and advise the reader to draw diagrams.

Case 1: $m = n = 1$.

Case 1-00: u_1 has no incoming positive edges, and u_3 has no outgoing positive edges. Add edges (u_3, t, u_1) and (u_1, t^{-1}, u_3). By the vertex creation lemma in section 3.2, the resulting recognizer S is equivalent to R. Obviously S is regular and lean. $\Delta(S) = \Delta(R) - 1$ as one deficiency has been repaired without creating any other deficiencies.

Case 1-11: u_1 has an incoming positive edge $e_0 = (u_0, T_\ell, u_1)$, and u_3 has an outgoing positive edge $e_3 = (u_3, T_p, u_4)$. Vertices u_1, u_2, and u_3 are all deficient. Since R is regular, we have $\ell, p > 1$. Split e_0 and e_3 into paths of the form

$$\langle(u_0, T_{\ell-1}, u), (u, s, u'), (u', t, u_1)\rangle, \ \langle(u_3, t, v), (v, s, v'), (v', T_{p-1}, u_4)\rangle.$$

The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Note t-irregular paths from u' to u_3 and from u_1 to v. Identify u' with u_3 and u_1 with v, so that the edges (u', t, u_1) and (u_3, t, v) become one edge (u_3, t, u_1), and the edges (u_1, t^{-1}, u') and (v, t^{-1}, u_3) become one edge (u_1, t^{-1}, u_3). The resulting recognizer S is regular, lean, and equivalent to R'. $\Delta(S) = \Delta(R) - 3$.

Case 1-01: u_1 has no incoming positive edges, and u_3 has an outgoing positive edge $e_3 = (u_3, T_p, u_4)$. Since R is regular, we have $p > 1$. Vertices u_2 and u_3 are deficient. Split e_3 as in case 1-11. The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Note a t-irregular path from u_1 to v. Identify u_1 with v, so that the edge (u_3, t, v) becomes (u_3, t, u_1) and the edge (v, t^{-1}, u_3) becomes (u_1, t^{-1}, u_3). The resulting recognizer S is regular, lean, and equivalent to R'. $\Delta(S) = \Delta(R) - 2$.

Case 1-10: u_1 has an incoming positive edge $e_0 = (u_0, T_\ell, u_1)$, and u_3 has no outgoing positive edge. This case is dual (and similar) to case 1-01.

Case 2: $m, n > 1$. Split e_1 and e_2 into paths of the form

$$\langle(u_1, T_{m-1}, u), (u, s, u'), (u', t, u_2)\rangle, \ \langle(u_2, t, v), (v, s, v'), (v', T_{n-1}, u_3)\rangle.$$

The resulting recognizer R' is regular, lean, and equivalent to R. $\Delta(R') = \Delta(R)$. Continue as in case 1-00, with t-edges (u', t, u_2) and (u_2, t, v) witnessing the deficiency of u_2.

Case 3: $m > 1$ and $n = 1$.

Case 3-0: u_3 has no outgoing positive edges. Split e_1 as in case 2. The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Continue as in case 1-00 with edges (u', t, u_2) and (u_2, t, u_3) witnessing the deficiency of u_2.

Case 3-1: u_3 has an outgoing positive edge $e_3 = (u_3, T_p, u_4)$.

Case 3-11: $p = 1$. If u_4 has no outgoing positive edges, then we have case 1-10 with the current u_2, u_3, and u_4 playing the role of u_1, u_2, and u_3 of case 1-10. If u_4 has an outgoing positive edge, then we have case 1-11 with the current u_2, u_3, and u_4 playing the role of u_1, u_2, and u_3 of case 1-11.

Case 3-12: $p > 1$. Vertices u_2 and u_3 are deficient. Split e_1 as in case 2, and split e_3 as in case 1-11. The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Note a t-irregular path from u' to v. Identify u' with v. The resulting recognizer S is regular, lean, and equivalent to R'. $\Delta(S) = \Delta(R') - 2$. ∎

COROLLARY 5.12 (deficit reduction). *There is a polynomial-time* deficit reduction algorithm *that transforms any lean regular recognizer into an equivalent zero-deficit lean regular recognizer*

5.3. Membership criterion and the reading algorithm.

The quasi-runs definition, the quasi-run labels corollary, and the tolerance definition of section 3.2 remain valid. By Part 1 of the syllables lemma in section 5.1, every partisan path that starts and ends with partisan edges is a quasi transition.

DEFINITION 5.13 (standard quasi transitions). *A quasi-transition q is* partisan *if it is a partisan path that starts and ends with partisan edges. A partisan quasi transition is* positive *(respectively,* negative*) if it is so as a path. A* neutral quasi transition *consists of one neutral edge. A quasi transition is* standard *if it is partisan or neutral.*

The label of every standard quasi transition is a syllable (rather than 1).

DEFINITION 5.14 (standard quasi runs). *A quasi-run $\langle q_1, \ldots, q_\ell\rangle$ is* standard *if every quasi-transition q_i is standard.*

LEMMA 5.15 (membership criterion). *Suppose that R is a zero-deficit lean regular recognizer, and let w be a reduced exponent word. The following are equivalent.*

1. *There is a standard quasi run with label w.*
2. *The group element w belongs to $\Gamma(R)$.*

Proof. $2 \to 1$: By the quasi-run labels corollary, the associate run of Q accepts a word equal to w, and so $w \in \Gamma(R)$.

$1 \to 2$: Suppose that the group element w belongs to $\Gamma(R)$. Without loss of generality, $w \neq 1$. By the recognizer's subgroup definition in section 2.2, R accepts an exponent word w' equal to w. Let π be a run that accepts w'.

We claim that the partisan and neutral edges alternate in π. To prove that, we assume that π has the form $\pi_1 + e + f + \pi_2$, where the edges e and f are both partisan or both neutral, and we prove that there is a shorter run accepting a word equal to w. Let v be the final vertex of e. If e and f are neutral, then by the regularity of R, $f = e^{-1}$, and so $\pi_1 + \pi_2$ is a shorter run accepting a word equal to w. So e and f are partisan. Let $T_m = \text{Label}(e)$ and $T_n = \text{Label}(f)$. Without loss of generality, $m > 0$; the case $m < 0$ is similar. If $n < 0$, then $f = e^{-1}$ because $\text{Fat}(t^{-1}, v) = 0$. Again, $\pi_1 + \pi_2$ is a shorter run accepting a word equal to w. Thus $n > 0$. Since $\Delta(R) = 0$, the vertex v is not deficient. It follows that $m = n = 1$, and there is an edge g such that $\langle e, f, g\rangle$ is a triangle. Then $\pi_1 + g^{-1} + \pi_2$ is a shorter run accepting an exponent word equal to w.

Notice that π is a standard quasi run. Let $Q = \langle q_1, \ldots, q_\ell\rangle$ be a standard quasi run with the fewest number of quasi transitions that tolerates an exponent word w_0 equal to w. Accordingly w_0 has the form $a_1^{p_1} \ldots a_\ell^{p_\ell}$. We show that w_0 is reduced. Recall the definition of reduced exponent words in section 5.1. Obviously w_0 satisfies the conditions (R1) and (R2). It remains to prove that it satisfies the condition (R3). It suffices to prove that the positive and negative quasi transitions alternate in Q.

By contradiction suppose that two partisan syllables σ_i and σ_{i+2} of the same sign are separated by a neutral syllable σ_{i+1}. Using the syllables lemma in section 5.1, replace the segment $\langle q_i, q_{i+1}, q_{i+2}\rangle$ with a single quasi transition whose quasi label equals $\sigma_i\sigma_{i+1}\sigma_{i+2}$ in G. This gives a quasi-run Q' with fewer quasi transitions that accepts a word equal to w, which contradicts the choice of Q. ∎

Lemma 5.16 (quasi transitions).

1. *Let R be a zero-deficit lean regular recognizer and u be a vertex of R. For every syllable σ, there is at most one standard quasi-transition q from u with quasi-label σ.*
2. *There is a polynomial-time algorithm that, given a zero-deficit lean regular recognizer R, a state u of R, and a syllable σ, determines whether there exists a standard quasi transition with profile of the form (u, σ, v) and, if yes, constructs the desired quasi transition.*

Proof. 1. The case $\sigma = s$ is obvious because R is regular. By contradiction assume that there exist distinct standard quasi transitions with the same quasi-label T_n from the same vertex u. We assume that $n > 0$; the case $n < 0$ is similar. Thus we have two distinct positive paths of the same length $2n - 1$ from u. Since neither path can be a prefix of the other, there is a vertex v where the two paths branch out which contradicts the leanness of R.

2. The case $\sigma = s$ is obvious. Thus we may assume that $\sigma = T_n$ for some $n \neq 0$. We assume $n > 0$; the case $n < 0$ is similar. Using the leanness of R, construct a unique empty or positive path π such that $\text{Length}(\pi) < 2n - 1$ but π cannot be extended to a longer positive path or such that $\text{Length}(\pi) \geq 2n - 1$ and π is the shortest such path. In the first case, the desired quasi transition does not exist. Consider the second case. If $\text{Length}(\pi) > 2n - 1$, then the desired quasi transition does not exists. But if $\text{Length}(\pi) = 2n - 1$, then we have the desired quasi transition. Notice that π may cycle from some point on. ∎

Lemma 5.17 (reading). *There is a polynomial-time* reading algorithm *that, given a lean regular recognizer R and a reduced exponent word w, determines whether the group element w belongs to $\Gamma(R)$.*

Proof. Taking into account the deficit corollary in section 5.2, we may assume without loss of generality that R is zero-deficit.

The reduced exponent word w is a concatenation $\sigma_1 \dots \sigma_\ell$ of syllables where the partisan syllables alternate with the neutral syllables and where, among the partisan syllables, positive syllables alternate with negative syllables. By the membership criterion lemma, it suffices to determine whether there exists a standard quasi-run $Q = \langle q_1, \dots, q_\ell \rangle$ with label w. By the quasi-transition lemma, there is at most one such quasi run.

Intuitively speaking, we use R to "read" w. Let $u_0 = o$. Suppose that $j \leq \ell$ and S has read the initial segment $\sigma_1 \dots \sigma_j$ of w and arrived at state u_j. In the process an initial segment $q_1 \dots q_j$ of the desired Q has been constructed.

If $j < \ell$, then apply the algorithm of the quasi-transition lemma to (R, u_j, σ_{j+1}). If the algorithm determines that there is no appropriate q_{j+1}, then $w \notin \Gamma(R)$. Otherwise the algorithm produces the appropriate q_{i+1}. Set u_{j+1} to be the final vertex of q_{j+1}, and proceed to read σ_{j+2}.

If $j = \ell$, then $w \in \Gamma(R)$ if and only if $u_\ell = o$. ∎

5.4. Path folding. This section is similar to section 3.4, but we need to do a little work to make the similarity apparent. The role of a fixed original letter a of section 3.4 is played by the word ts in this section. The fact that the word ts is not a single letter creates some difficulties.

Consider a regular recognizer R. Since R is regular, every vertex of R has at most one outgoing s-edge.

Definition 5.18 (dual vertices). *If a vertex u has an outgoing s-edge e, then the end of e is the* dual *of the vertex u; otherwise, u is an s*-orphan.

It will be convenient to pretend that orphans have dual vertices as well.

Definition 5.19 (virtual duals). *With every orphan u, we associate an object v outside of R in such a way that distinct objects are associated with distinct vertices. These objects v are called* virtual elements *of R. The nature of virtual elements is of no importance. If an orphan u is associated with a virtual element v, we say that u and v are the* duals *of each other. Further, the triples (u, s, v) and (v, s, u) are called* virtual edges *of R.*

Recall that a partisan path has the following properties: partisan and neutral edges alternate and all partisan edges have the same sign.

Definition 5.20 (regular paths). *A* regular path *π is a nonempty partisan path of the form $\langle e_1, f_1, \ldots, e_k, f_k\rangle$ subject to the following restrictions.*

- *Edges e_i are partisan.*
- *Edges f_i are neutral, and the final edge f_k may be virtual.*
- *The initial vertices of edges e_i are all distinct.*

If f_k is virtual, then π is odd, *and if f_k is real, then π is* even. *Notice that the sequence $\langle e_1, \ldots, e_k\rangle$ completely determines the regular path π which will be denoted* $\mathrm{RP}(e_1, \ldots, e_k)$.

Definition 5.21 (even vertices). *Let π be a regular path $\langle e_1, f_1, \ldots, e_k, f_k\rangle$, where $e_i = (u_{i-1}, \sigma_i, v_i)$ and $f_i = (v_i, s, u_i)$. Notice that every* $\mathrm{Length}(\pi[u_0, u_i])$ *is even and every* $\mathrm{Length}(\pi[u_0, v_i])$ *is odd. Call vertices u_i* even *and vertices v_i* odd. *So the even vertices are the initial vertices of the partisan edges and the final vertices of the neutral edges. The* even vertex sequence *of π is the sequence $\langle u_0, \ldots, u_{k-1}, u_k\rangle$. By the definition of regular paths, even vertices $u_0, \ldots, u_{k-1}$ are distinct.*

In the following definition, we redefine the notion of branch and cycle and some related notions to the case of regular paths. We will not apply the old versions of these notions to regular paths. In fact, the old notions will not be used in the rest of this paper, except that here and there we refer the reader to previous sections where the old notions are in use.

Definition 5.22 (branches, cycles, etc.). *Consider a regular path $\pi = \langle e_1, \ldots, e_k\rangle$ with even vertex sequence $\langle u_0, u_1, \ldots, u_k\rangle$.*

- *π is a* branch *if $u_k \notin \{u_0, \ldots, u_{k-1}\}$.*
- *π is an* impasse *if it is a branch and u_k does not have an outgoing partisan edge of the sign of π.*
- *π is a* cycle at u_0 *if $u_k = u_0$.*
- *π is a* noose *if $u_k = u_i$ for some positive $i < k$.*
- *If π is a noose and $u_k = u_i$, where $i < k$, then π splits into the* loop $\mathrm{RP}(e_{i+1}, \ldots, e_k)$ *and the* tail *$\langle e_1, \ldots, e_i\rangle$ of the noose.*
- *π is* closed *if it is an impasse, a cycle, or a noose.*

Lemma 5.23 (closed paths). *Every partisan edge e gives rise to a closed regular path $\pi = \mathrm{RP}(e_1, \ldots, e_k)$ with $e_1 = e$. Furthermore, there is a polynomial-time algorithm that, given (a regular recognizer and) a partisan edge e, constructs such a path π.*

Proof. The desired algorithm is iterative. At the first round, it constructs the regular path $\mathrm{RP}(e_1)$ with $e_1 = e$. Suppose that we did i rounds and constructed a regular path $\mathrm{RP}(e_1, \ldots e_i)$. If it is closed, then halt. Otherwise $\mathrm{RP}(e_1, \ldots e_i)$ is even. Let e_{i+1} be the lexicographically first partisan edge from u_i of the sign of e, and construct $\mathrm{RP}(e_1, \ldots e_{i+1})$. The process converges because the number of vertices is finite. ∎

DEFINITION 5.24 (edge splitting). *Suppose that p, q, and r are positive integers such that $r = p + q$. By the syllables lemma in section* 5.1, *$T_r = T_p s T_q$, and $T_{-r} = T_{-p} s T_{-q}$. To* split an edge $e = (u, T_r, v)$ according to (p, q), *create two new vertices u' and v' and replace edges e and e^{-1} with six new edges: (u, T_p, u'), (u', s, v'), and (v', T_q, v) and their inverses. Splitting an edge $e = (u, T_{-r}, v)$ according to (p, q) is defined similarly; just substitute T_p, T_q, and T_r with T_{-p}, T_{-q}, and T_{-r}, respectively.*

LEMMA 5.25 (edge splitting). *Edge splitting preserves the regularity of the recognizer, the recognizer subgroup, and the amount of fat of the recognizer.*

Proof. The proof is obvious. ∎

DEFINITION 5.26 (vertex creation). *Let $\pi = \mathrm{RP}(e_1, \ldots, e_k)$, ν be the initial vertex of π, and m be a positive even number such that $m <$ Length(π) and there is no even vertex of π with* Length$(\pi[\nu, u]) = m$. *We explain how to create, on π, a new even vertex v' at distance m from ν as well as its dual u' at distance m from ν. Let $L(i) =$ Length$(\mathrm{RP}\langle e_1, \ldots, e_i\rangle)$ for $i = 0, \ldots, k$. Find the index i with $L(i-1) < m < L(i)$, and split the edge e_i according to (p, q), where $p = (m - L(i-1))/2$ and $q = (L(i) - m)/2$. This creates, on π, the desired even vertex v' as well as its dual u'.*

COROLLARY 5.27 (vertex creation). *Vertex creation preserves the regularity of the recognizer, the recognizer subgroup, and the amount of fat of the recognizer.*

We adjust the entanglement definition of section 3.4 to fit our needs in this section. The two path divisor definition of section 3.4 remains valid.

DEFINITION 5.28 (entanglement). *Suppose that π and ρ are regular paths of the same sign and with the same initial vertex ν. Suppose further that either path is a branch or a cycle. The two paths are* entangled *if there exist even vertices u and v on π and ρ, respectively such that $u \neq v$ and*

$$\mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho).$$

Otherwise the two paths are disentangled.

The entanglement algorithm corollary of section 3.4 remains valid.

LEMMA 5.29 (entanglement). *If π and ρ are entangled and even vertices u and v witness the entanglement, then the identification of u and v does not change the recognizer subgroup.*

Proof. Let $d = \mathrm{Div}(\pi, \rho)$, $2k = \mathrm{Length}(\pi[\nu, u])$, $2\ell = \mathrm{Length}(\rho[\nu, v])$, and $H = \mathrm{Coset}(\nu)$. We assume that π are ρ are positive; the negative case is similar. By the syllables lemma in section 5.1, Label$(\pi[\nu, u]) = (ts)^k$, and Label$(\rho[\nu, v]) = (ts)^\ell$. By the coset stability lemma in section 2.4, Coset$(u) = H(ts)^k$ and Coset$(v) = H(ts)^\ell$. By the vertex identification lemma in section 2.4, it suffices to prove that $H(ts)^k = H(ts)^\ell$. Since u and v witness the entanglement, we have $2k = 2\ell \mod d$.

Case 1: Both paths are branches. Then $k = \ell$, and therefore $H(ts)^k = H(ts)^\ell$.

Case 2: One of the paths is a branch and the other is a cycle. Without loss of generality, π is a cycle, and ρ is a branch. Then $d = 2k$, $H(ts)^k = H$, and $2\ell = p \cdot 2k$ for some integer p. Then $\ell = kp$, and $H(ts)^\ell = H((ts)^k)^p = H = H(ts)^k$.

Case 3: Both paths are cycles. Then $d = \text{g.c.d.}(2k, 2\ell)$, and $H(ts)^k = H(ts)^\ell = H$. Clearly d is even; let $\delta = d/2$ so that $\delta = \text{g.c.d.}(k, \ell)$. Since $\delta = \text{g.c.d.}(k, \ell)$, there are integers i and j such that $ik + j\ell = \delta$, and so $H(ts)^\delta = H((ts)^k)^i((ts)^\ell)^j = H$. Since $2k = 2\ell \mod d$, there is an integer p such that $2k = pd + 2\ell$, and therefore $k = p\delta + \ell$. Then $H(ts)^k = H((ts)^\delta)^p(ts)^\ell = H(ts)^\ell$. ∎

DEFINITION 5.30 (even vertex disjoint regular paths). *Consider two regular paths sharing the same initial vertex ν. The two paths are* even vertex disjoint off ν *if ν is the only even vertex on both paths.*

If two regular paths are even vertex-disjoint off their common initial vertex ν, then the dual of ν is the only possible odd vertex on both paths. But it is possible that an even vertex of one path appears as an odd vertex on the other path.

DEFINITION 5.31 (path folding). *Suppose that π and ρ are regular paths such that*

- *they have the same sign and the same initial vertex ν,*
- *either path is a branch or a cycle,*
- $\text{Length}(\pi) \geq \text{Length}(\rho)$ *if both paths are branches, and*
- $\text{Length}(\pi) = \text{Div}(\pi, \rho)$ *if both paths are cycles.*

To fold ρ into π, *execute the following* folding algorithm:

1. *Apply the entanglement algorithm to π and ρ. If the number of vertices is reduced, then halt. Otherwise the paths are disentangled.*
2. *For each even vertex v on ρ, create, on π, a new even vertex v' and its dual such that*
$$\text{Length}(\pi[\nu, v']) = \text{Length}(\rho[\nu, v]) \mod \text{Div}(\pi, \rho)$$
unless π has an even vertex at this position already.
3. *Identify every clone v' with its original v, and identify the dual of v' with the dual of v.*
4. *Remove all partisan edges of ρ and their inverses.*

The path folding lemma of section 3.4 remains valid. Its formulation does not change at all, and its proof requires only small adjustments. For the reader's convenience we give here all details.

LEMMA 5.32 (path folding). *Let π and ρ be as in the path folding definition. Folding ρ into π preserves the recognizer subgroup. If the algorithm halts at stage* 1, *then the number of vertices of the recognizer decreases. Otherwise the number of vertices is unchanged, and the (amount of) fat changes as follows.*

(BB) *Suppose that π and ρ are branches of lengths m and n, respectively.*
If $m = n$, then the fat decreases by 2.
If $m > n$ and ρ is an impasse, then the fat decreases by 1.
If $m > n$ but ρ is not an impasse, then the fat does not change.

(CB) *Suppose that π is a cycle and ρ is a branch.*
If ρ is an impasse, then the fat decreases by 1*;*
otherwise, the fat does not change.

(CC) *Suppose that π and ρ are cycles. Then the fat decreases by* 2.

Proof. We consider only the case in which π and ρ are positive; the case in which they are negative is similar.

Let R be the given recognizer, and let R_p be the recognizer obtained from R by executing p stages of the folding algorithm, so that $R_0 = R$. Let the even vertex sequence of π be $\langle u_0, \ldots, u_k \rangle$. Let $\rho = \text{RP}(f_1, \ldots, f_\ell)$ and the even vertex sequence of ρ be $\langle v_0, \ldots, v_\ell \rangle$. Let $d = \text{Div}(\pi, \rho)/2$. The case in which π and ρ are entangled is obvious. Assume that π and ρ are disentangled. Then nothing happens at stage 1, and so $R_1 = R_0$.

First we note that the vertices of R_4 are those of R_1. Indeed all vertices v'_j and their duals created at stage 2 are identified with the respective vertices v_j and their duals at stage 3; thus, the vertices of R_3 are those of R_1. And the vertices do not change at stage 4.

Second we show that $\Gamma(R_4) = \Gamma(R)$. By the vertex creation lemma in section 2.4, $\Gamma(R_2) = \Gamma(R_1)$. By the vertex identification lemma in section 2.4, $\Gamma(R_3) = \Gamma(R_2)$. It remains to show that $\Gamma(R_4) = \Gamma(R_3)$. Let $n_j = \frac{1}{2}\text{Length}(\pi[v_0, v_j])$, $r_j = n_j \mod d$, and $v'_0 = u_0$. It suffices to show that, for every partisan edge f_j of ρ, R_4 has a path P with the profile of f_j. The profile of f_j is (v_{j-1}, T_p, v_j), where $p = n_j - n_{j-1}$. In scenario (BB), the desired path P is $\pi[v'_{j-1}, \mathrm{Dual}(v'_j)]$. In scenarios (CB) and (CC), $p = d \cdot q + (r_{j+1} - r_j)$ for some q. The desired path P starts at v'_{j-1} and ends at $\mathrm{Dual}(v'_j)$. If $r_i \leq r_{i+1}$, then π does q full revolutions around π. If $r_i > r_{i+1}$, then $q > 0$, and π does $q-1$ full revolutions around π.

Finally we prove the claims about the fat. By the vertex creation lemma, $\mathrm{Fat}(R_2) = \mathrm{Fat}(R_1)$. Thus we need only to examine the evolution of the fat from R_2 to R_4. Furthermore, it suffices to examine the evolution of the numbers $\mathrm{Fat}(t, v_j)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_j))$. Indeed, v'_j and its dual merge with v_j and its dual, respectively, at stage 3. $\mathrm{Fat}(t^{-1}, v_j)$ and $\mathrm{Fat}(t, \mathrm{Dual}(v_j))$ do not change. And if a vertex v of R_2 differs from any v_j, from any v'_j, and from their duals, then the immediate vicinity of v does not change.

If $0 < j < \ell$, then $\mathrm{Fat}(t, v_j)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_j))$ do not change from R_2 to R_4. Indeed, as a result of identification with v'_j, at stage 3, the vertex v_j acquires one outgoing positive edge, and $\mathrm{Dual}(v_j)$ acquires one outgoing negative edge, but then, at stage 4, v_j loses one outgoing positive edge, namely, f_{j+1}, and $\mathrm{Dual}(v_j)$ loses one outgoing negative edge, namely, f_j^{-1}.

The vertices v_0 and its dual do not acquire any outgoing edges at stage 3, and thus neither $\mathrm{Fat}(t, v_0)$ nor $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ increase on stage 3. v_0 loses one positive outgoing edge, namely, f_0, at stage 4, and so $\mathrm{Fat}(t, v_0)$ decreases by 1 from R_2 to R_4. $\mathrm{Dual}(v_0)$ does not lose any negative outgoing edge in scenarios (BB) or (CB), and so $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ does not change in scenarios (BB) and (CB). Since $v_0 = v_\ell$ in scenario (CC), it remains only to examine the evolution of the numbers $\mathrm{Fat}(t, v_\ell)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ in the three scenarios.

(BB) In this scenario, we first suppose that $m = n$. Since π and ρ are disentangled, $u_k = v_\ell$. The vertices v_ℓ and its dual do not acquire any outgoing edges at stage 3 and lose only one outgoing edge, namely, the negative edge f_l^{-1}, at stage 4. Thus $\mathrm{Fat}(t, v_\ell)$ does not change, and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ decreases by 1. To summarize, $\mathrm{Fat}(t, v_0)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ decrease by 1 while $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ and $\mathrm{Fat}(t, v_\ell)$ do not change. Hence $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$.

Second we suppose that $m > n$. At stage 3, v_ℓ acquires one positive outgoing edge, and $\mathrm{Dual}(v_\ell)$ acquires one negative outgoing edge. At stage 4, v_ℓ loses no outgoing edges while $\mathrm{Dual}(v_\ell)$ loses f_ℓ^{-1}. Thus $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ do not change. If ρ is not an impasse, then $\mathrm{Fat}(t, v_\ell)$ increases by 1; otherwise, $\mathrm{Fat}(t, v_\ell)$ remains zero throughout the process. We summarize. If ρ is an impasse, then $\mathrm{Fat}(t, v_0)$ decreases by 1 while $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$, $\mathrm{Fat}(t, v_\ell)$, and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 1$. If ρ is not an impasse, then $\mathrm{Fat}(t, v_0)$ decreases by 1, $\mathrm{Fat}(t, v_\ell)$ increases by 1, and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2)$.

(CB) This scenario is similar to the case $m > n$ of scenario (BB). Let us just point out that the vertex v_ℓ does not occur on π in R. Indeed suppose the opposite. Since π and ρ are even vertex disjoint off their common initial vertex and $u_k = u_0$, we have $v_\ell = u_0 = v_0$. But then ρ is a cycle which contradicts scenario (CB).

(CC) In this scenario, $v_0 = v_\ell$, and thus $\mathrm{Fat}(t, v_\ell)$ decreases by 1. $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ does not change at stage 3 and decreases by 1 at stage 4 because f_ℓ^{-1} is removed. To summarize, the overall change in the fat is this: both $\mathrm{Fat}(t, v_\ell)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ decrease by 1. Thus $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$. $\square$

5.5. Weight reduction algorithm. The recognizer weight definition of section 3.5 remains in force.

Definition 5.33 (recognizer weight). *The* weight *of a recognizer R is a pair (i, j) of natural numbers, where i is the number of vertices of R and $j = Fat(R)$. The weights are ordered lexicographically with the number of vertices being the more significant component.*

Lemma 5.34 (weight reduction). *There is a polynomial-time* weight reduction algorithm *that reduces any recognizer to an equivalent lean regular recognizer.*

Proof. The proof is very close to the proof of the weight reduction lemma of section 3.5. There is a slight difference in the beginning, and so we give here that new beginning.

We construct an iterative algorithm that transforms the given recognizer by means of path folding; the algorithm halts when the recognizer is lean. By the folding lemma of section 5.4, the algorithm preserves the recognizer subgroup.

We describe one round of the algorithm and show that the weight decreases at each round. It will be obvious that the algorithm is polynomial time.

If the current recognizer R is regular and lean, halt. If there exists an irregular path, then identify the two ends of the path. This decreases the recognizer weight. Irregular paths were defined in section 4.

If there is a vertex ν with $\mathrm{Fat}(\nu) > 0$, find a witness $(t^\varepsilon, \nu, e_1, f_1)$ for this fact, where $\varepsilon \in \{1, -1\}$, ν is a vertex with $\mathrm{Fat}(t^\varepsilon, \nu) > 0$, and e_1 and f_1 are two t^ε-edges from ν of the sign of ε. We consider only the case $\varepsilon = 1$; the case $\varepsilon = -1$ is similar. Use the algorithm of the closed paths lemma to construct closed regular paths E and F continuing the e_1 and e_2, respectively.

The rest of the proof mimics the corresponding part of the proof of the weight reduction lemma of section 3.5. ∎

5.6. The theorem.

Theorem 5.35. *There is a polynomial-time decision algorithm for the membership problem for the succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. More explicitly, there is an algorithm such that*

(i) *given exponent words $h_1, \dots, h_m$ and w, the algorithm decides whether the subgroup H generated by $h_1, \dots, h_m$ contains w, and*

(ii) *the algorithm runs in time polynomial in $|h_1| + \cdots + |h_m| + |w|$.*

Proof. We describe the desired decision algorithm. Use the construction algorithm of section 2.2 to construct a recognizer R_1 for H. Use the weight reduction algorithm of section 5.5 to reduce R_1 into a lean regular recognizer R_2. Use the reading algorithm of section 5.3 to check whether the group element w belongs to H. Since all constituent algorithms are polynomial time, the decision algorithm is polynomial time. ∎

6. Main theorem. We reiterate the main theorem formulated in section 1.

Theorem 6.1 (main). *The membership problem for the modular group* $\mathrm{PSL}_2(\mathbb{Z})$, *with integer entries in the standard decimal notation, is polynomial-time decidable.*

Proof. In the previous section, we proved the polynomial-time decidability of the membership problem for the succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. Thus it suffices to prove that the membership problem for the modular group is polynomial-time reducible to the membership problem for $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. We construct the desired reduction.

By the modular group as a free product proposition in section 1, the modular group is isomorphic to $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$, where

$$s = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \text{and} \qquad t = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}.$$

Recall that a matrix is identified with its negative in the modular group.

Consider the four basic elementary transformations over the columns of unimodular matrices:

- subtract the first column from the second,
- subtract the second column from the first,
- add the second column to the first, and
- add the first column to the second.

Applying these transformations to the identity matrix, we get the basic elementary matrices

$$\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix} \quad \begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix} \quad \begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix} \quad \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}.$$

Multiplying a unimodular matrix M on the right by a basic elementary matrix E performs the corresponding column operation on M, and multiplying M by a E^k performs the column operation k times. Check that

$$st = \begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}, \quad st^{-1} = \begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}, \quad ts = \begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}, \quad t^{-1}s = \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}.$$

We need to transform an arbitrary unimodular matrix M into an exponent word that represents M. Note that every exponent word w represents a unimodular matrix and thus gives rise to an operation $X \mapsto X \times w$ over unimodular matrices. It suffices to construct a polynomial-time procedure that reduces any unimodular matrix

$$M = \begin{pmatrix}a & b\\ c & d\end{pmatrix}$$

to the identity matrix by means of such operations.

Without loss of generality $a > 0$, because we can work either with M or with $-M$. To simplify exposition (though not the algorithm), we can assume that $b > 0$ as well. Indeed, if $b < 0$, then replace M with $M \times (t^{-1}s)^k$, where k is the least integer such that $ka > |b|$.

If $a \geq b$, then compute the number k such that $0 \leq a - kb < b$, and replace M with $M \times (st^{-1})^k$, so that we have $a \mod b$ in the left upper corner of the resulting matrix. Note that $a \mod b < a/2$. Indeed, if $b \leq a/2$, then $a \mod b < b \leq a/2$; otherwise, $b > a/2$, and $a \mod b \leq a - b < a/2$. Similarly, if $a < b$, then compute the number k such that $0 \leq b - ka < a$, and replace M with $M \times (st)^k$, so that we have $b \mod a$ in the right upper corner of the resulting matrix. We have $b \mod a \leq b/2$.

Keep doing that until you have zero in one of the upper corners. In every two steps, the entries in both upper corners are more than halved. It follows that this iteration works in linear time.

Thus, without loss of generality, we may assume that the left upper entry a of the given matrix M is zero; the case when the right upper entry b is zero is similar. So

$$M = \begin{pmatrix}0 & b\\ c & d\end{pmatrix}.$$

Further, we may assume without loss of generality that $b = 1$ and $c = -1$ because the determinant of M is 1, and we can work with either M or its negative. Replace M with $(M \times t^{-1}s)^d$; the result is the matrix

$$s = \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}.$$

Finally replace the resulting matrix s by $s \times s$, and get (the negative of) the identity matrix. $\square$

REFERENCES

[1] A. Blass and Y. Gurevich, *Matrix transformation is complete for the average case*, SIAM J. Comput., 22 (1995), pp. 3–29.

[2] J.-Y. Cai, W. H. J. Fuchs, D. Kozen, and Z. Liu, *Efficient Average-case algorithms for the modular group*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1994, pp. 143–152.

[3] Y. Gurevich, *Average case completeness*, J. Comput. System Sci., 42 (1991), pp. 346–398.

[4] Y. Gurevich, *Matrix decomposition problem is complete for the average case*, in Proceedings of the 31st Annual Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1994, pp. 802–811.

[5] I. Kapovich and A. Miasnikov, *Stallings foldings and subgroups of free groups*, J. Algebra, 248 (2002), pp. 608–668.

[6] L. Levin, *Average case complete problems*, SIAM J. Comput., 15 (1986), pp. 285–286.

[7] R. Lyndon and P. E. Schupp, *Combinatorial Group Theory*, Springer-Verlag, Berlin, 1977.

[8] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations*, Dover, New York, 1976.

[9] P. E. Schupp, *Coxeter groups, 2-completion, perimeter reduction and subgroup separability*, Geom. Dedicata, 96 (2003), pp. 179–198.

[10] J. R. Stallings, *Topology of finite graphs*, Invent. Math., 71 (1983), pp. 551–565.