Discrete & Computational Geometry (2023) 70:323–354
https://doi.org/10.1007/s00454-022-00411-x

Algorithms for Contractibility of Compressed Curves on 3-Manifold Boundaries

Erin Wolf Chambers[1] · **Francis Lazarus**[2] · **Arnaud de Mesmay**[3] · **Salman Parsa**[1]

Received: 3 June 2021 / Revised: 23 March 2022 / Accepted: 3 April 2022 /
Published online: 18 July 2022

Abstract

In this paper we prove that the problem of deciding contractibility of an arbitrary closed curve on the boundary of a 3-manifold is in **NP**. We emphasize that the manifold and the curve are both inputs to the problem. Moreover, our algorithm also works if the curve is given as a compressed word. Previously, such an algorithm was known for simple (non-compressed) curves, and, in very limited cases, for curves with self-intersections. Furthermore, our algorithm is fixed-parameter tractable in the size of the input 3-manifold. As part of our proof, we obtain new polynomial-time algorithms for compressed curves on surfaces, which we believe are of independent interest. We provide a polynomial-time algorithm which, given an orientable surface and a compressed loop on the surface, computes a canonical form for the loop as a compressed word. In particular, contractibility of compressed curves on surfaces can be decided in polynomial time; prior published work considered only constant genus surfaces. More generally, we solve the following normal subgroup membership problem in polynomial time: given an arbitrary orientable surface, a compressed closed curve γ, and a collection of disjoint normal curves Δ, there is a polynomial-time algorithm to decide if γ lies in the normal subgroup generated by components of Δ in the fundamental group of the surface after attaching the curves to a basepoint.

Editor in Charge: Kenneth Clarkson

Erin Wolf Chambers: This work was funded in part by the National Science Foundation through grants CCF-1614562, CCF-1907612 and DBI-1759807.
Francis Lazarus: This author is partially supported by the French ANR projects GATO (ANR-16-CE40-0009-01) and MINMAX (ANR-19-CE40-0014) and the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.
Arnaud de Mesmay: This author is partially supported by the French ANR projects ANR-17-CE40-0033 (SoS), ANR-16-CE40-0009-01 (GATO) and ANR-19-CE40-0014 (MINMAX).
Salman Parsa: This work was funded in part by the National Science Foundation through grant CCF-1614562 as well as funding from the SLU Research Institute.

Extended author information available on the last page of the article

Keywords 3-Manifolds · Surfaces · Computational topology · Contractibility · Compressed curves

Mathematics Subject Classification 57M05 · 68R99

1 Introduction

In a topological space X, a closed curve $c\colon \mathbb{S}^1 \to X$ is *contractible* if the map c extends to a map of the standard disk, $f\colon D \to X$, $f|_{\partial D} = c$; such a curve can be continuously deformed to a point. Dating back to Dehn [10] more than a century ago, the problem of testing contractibility has been an important focus in the development of efficient algorithms in computational topology and group theory. Such problems are now well understood in two dimensions, with recent works of the second author and Rivaud [30] and Erickson and Whittlesey [16], providing linear-time algorithms to test contractibility and free homotopy of curves on surfaces; Despré and the second author [11] also developed efficient algorithms for the closely related problem of computing the geometric intersection number of curves. On the other hand, in higher dimensions, homotopy problems quickly become undecidable: testing contractibility of curves is already undecidable on 2-dimensional simplicial complexes and in 4-manifolds (see Stillwell [42, Chap. 8]).

The aim of this paper is to further improve our understanding of the intermediate 3-dimensional case. In a 3-manifold, testing whether a curve is contractible is known to be decidable, but the best known algorithms are intricate and rely on the proof of the Geometrization Conjecture. We refer to the survey of Aschenbrenner et al. [2]. When the input manifold is a knot complement, this problem contains as a special case the famous Unknot Recognition problem, which asks whether an input knot in $\mathbb{R}^3$ is trivial. Hass et al. [19] showed that this problem is in **NP** and Lackenby [29] proved that the problem to be also in co-**NP**. Recently, Colin de Verdière and the fourth author [8] initiated the study of a problem in between those two, which is testing the contractibility of a closed curve γ which lies on the boundary of a triangulated 3-manifold M, and they provided an algorithm to solve this problem in time $2^{O((|M|+|\gamma|)^2)}$. Here, $|M|$ and $|\gamma|$ respectively refer to the number of tetrahedra in the triangulated 3-manifold M and to the number of line segments that make up γ. They asked whether the problem lies in the complexity class **NP** and proved that it holds in two cases: when the number of self-intersections of γ is at most logarithmic, and when the boundary surface is a torus. We remark that the existence of an algorithm to test contractibility of a closed curve on the boundary actually follows from the earlier results of Waldhausen [43], and an exponential-time algorithm can be derived from the standard 3-manifold literature, as we will explain later in this paper (cf. Sects. 1.2 and 3). However, the algorithm of [8, 9] is simpler in the sense that it relies only on the proof of the Loop Theorem.

Our main result is that the problem of contractibility when the curve is on the boundary of the 3-manifold is in **NP**. Furthermore, the algorithm that we provide has two additional features. First, it is fixed parameter tractable in the size of the input triangulation: for a given triangulated 3-manifold M, after a preprocessing taking exponential time in $|M|$, we can solve any contractibility test for curves on the boundary in polynomial time. Second, our algorithm also works if the input curve is *compressed* via a straight-line program, in particular, it can be exponentially longer than its encoding size (we defer to Sect. 2 for the precise definitions).

Theorem 1.1 *Let a triangulated* 3*-manifold M and a curve γ on the boundary of M be given as the input. The problem of deciding whether γ is contractible in M is in* ***NP****, and there is a deterministic algorithm that solves the problem in time* $2^{O(|M|^3)}\,\mathrm{poly}(|\gamma|)$. *This is also the case if γ is given as a compressed word, i.e., a straight-line program.*

Our proof of Theorem 1.1 will reduce the problem to another problem involving only curves on surfaces. However, there is an important subtlety. Many algorithms in 3-dimensional topology involve an exponential blow-up in the size of the objects under consideration. This is the case for example for UNKNOT RECOGNITION, where the classical algorithm looks for a disk spanning the knot, but this disk might be exponentially large, as shown by Hass et al. [20]. In order to get **NP** membership, this exponential blow-up is controlled using a compressed[1] system of coordinates called *normal coordinates* (see Sect. 2 for definitions). Our approach follows a similar scheme and suffers from a similar blow-up, which forces us to deal with normal curves. Precisely, we show that the problem of contractibility of an arbitrary curve on the boundary of a 3-manifold reduces to the following problem.

> Given a family $\varDelta$ of disjoint normal closed curves on a triangulated surface S, and a curve γ on S, decide if γ lies in the normal subgroup N of the fundamental group of S determined by the curves of $\varDelta$ (appropriately connected to the basepoint).

This is equivalent to deciding if the curve γ is contractible in the space resulting from S by gluing a disk to each component of $\varDelta$. We call this problem the *disjoint normal subgroup membership* problem, and we provide an algorithm to solve it in polynomial time, despite the highly compressed nature of $\varDelta$ and γ.

Theorem 1.2 *Let an orientable triangulated surface S, a closed normal multi-curve $\varDelta$, and a compressed curve γ be given as the input. There is a polynomial-time algorithm for deciding the disjoint normal subgroup membership problem for γ and $\varDelta$ on S.*

To solve the disjoint normal subgroup membership problem, one of our technical tools is a polynomial-time algorithm to test triviality of a compressed closed curve on a surface, which might be of independent interest. In fact, we compute a canonical form for such a curve, in the sense that there is a unique canonical representative in each based homotopy class.

Theorem 1.3 *Let an orientable triangulated surface S and a compressed loop γ on S described by a straight-line program be given. Then one can transform γ, in polynomial time, into a canonical form in its based homotopy class, given as a compressed word. In particular, we can test in polynomial time whether γ is contractible.*

[1] There are two different forms of compression going on in this paper: normal coordinates and straight line programs. We keep the name *compressed* for the latter, while a curve or a multicurve encoded with normal coordinates will be simply called a *normal* (*multi-*)*curve*.

We emphasize that in Theorem 1.3 the surface S is part of the input. The analogous theorem, if we assume that surface S is not part of the input, has been known in a much broader context, as we discuss in the next section.

1.1 Related Work

1.1.1 Algorithms in 3-Manifold Topology

There is now a large body of research on the computational complexity of problems in 3-manifold topology and knot theory. To paint a picture in broad strokes, topological problems about 3-manifolds are typically decidable, using a combination of geometric, algebraic and topological tools and the complexity of the best known algorithms range from exponential [1, 19] to quite galactic, see for example [26] for the homeomorphism problem.[2] In contrast, only very few complexity lower bounds are known [1, 4, 27, 35] and for many problems, including ours, it is open whether a polynomial-time algorithm exists. One particular tool that our work relies on is *normal surface theory*, which provides a powerful framework to enumerate and analyze the topologically relevant embedded surfaces in a 3-manifold. Actually, tools from normal surface theory [41] provide an off-the-shelf algorithm proving that deciding contractibility of a *simple* curve on the boundary of a 3-manifold is in **NP**, as explained in [8, Sect. 3]. However, normal surface theory is ill-adapted to study surfaces that are not embedded [7], hence the more technical path that we follow in this paper. We refer to [28] for a recent and extensive survey on algorithms in 3-manifold topology.

1.1.2 Geometric and Combinatorial Group Theory

The problem of deciding the contractibility of a curve in a topological space, which is given for example as a finite simplicial complex, can be equivalently rephrased as a problem of testing the triviality of a word in a finitely presented group, the fundamental group of the space. In the case of 2- and 3-manifolds, the geometry or topology of the underlying manifold has a strong impact on the properties of the group, and there is a vast amount of literature on this interaction, see for example [18]. We refer to [2, 3] for the case of 3-manifold groups. Of particular interest for the word problem is the notion of a *Dehn function*, which upper bounds the area of a disk certifying the triviality of a word, and thus controls the complexity of a brute-force algorithm to solve the word problem. For 3-manifold groups, this function can be exponential [14, Thm. 8.1.3], making such an approach at least exponentially worse than ours. When the Dehn function of a group presentation is polynomial (as is the case, for instance, for the fundamental group of a compression body; being isomorphic to the free product of a surface group with a free group, its Dehn function is indeed linear), it can be used to prove that the word problem is in **NP**. However, the word problem is defined for a fixed presentation. In our problem, the group presentation itself is part of the input, and we cannot use the bound on the Dehn function directly.

[2] We recall, however, that the homeomorphism problem is undecidable in dimensions four or higher.

1.1.3 Algorithms for Compressed Curves and Surfaces

As we hinted at in the introduction, 3-dimensional algorithms naturally tend to involve exponentially large objects, which are generally compressed using normal coordinates. This incurs a need for efficient algorithms dealing with normal curves and surfaces, even for the most basic tasks. For example, it is not easy to test whether a curve described by normal coordinates is connected. By now, there are many known techniques to handle topological problems on compressed curves, see for example works of Agol et al. [1], Schaefer et al. [38, 39], Erickson and Nayyeri [15], Bell [5], or Dynnikov [12] and Dynnikov and Wiest [13]. None of these techniques seem to directly solve our disjoint normal subgroup membership problem, but we rely extensively on the algorithm of Erickson and Nayyeri [15] as a subroutine (see Sect. 4). However, normal coordinates do not describe curves with self-intersections, which are our main object of interest in this paper. This is why we rely on a different compression model in the form of *straight-line programs*. The use of such programs in topology or geometric group theory is not new. As mentioned above, [38, 39] already used them for dealing with normal curves on surfaces, while there is a sketch of an algorithm to test triviality of compressed curves on a surface of genus 3 in an appendix of Schleimer [40, App. A]. His approach seems to be readily generalizable to higher genus, but the dependency on the genus is not made explicit. More recently, Holt et al. [23] provided a polynomial-time algorithm to test triviality of compressed words in *hyperbolic groups*, of which (most) surface groups are a subclass. The difference with our Theorem 1.3 is that in their result, the group presentation is not part of the input. While both approaches could plausibly be used in our setting, we rely on different tools to provide a complete proof of Theorem 1.3: our approach treats the surface as an input and works for any genus, and seamlessly generalizes to the case of wedges of surfaces that we also need.

1.2 Summary of Our Techniques

The standard methods of normal surface theory allow us to reduce our main contractibility problem to the case where the input manifold belongs to a particular class of manifolds called *compression bodies*, see the proof of Proposition 3.1. The fundamental group of a compression body is roughly a free product of surface groups, and thus one can rely on known algorithms for surfaces [16, 30] to solve the contractibility problem there. However, due to the exponential size of normal surfaces, this would only yield an exponential time algorithm. In order to improve on this, we identify the precise problem that we need to solve to be the disjoint normal subgroup membership problem and set out on a quest to solve it efficiently. While an **NP** algorithm would suffice for our main result, we will develop a polynomial-time algorithm.

The disjoint normal subgroup membership problem is made delicate by the compressed nature of the curves in Δ. If they were given as disjoint curves, say, on the 1-skeleton of the surface, we could glue a disk on each of them, and observe that the resulting complex is homotopically equivalent to a wedge of surfaces,[3] allowing us to reduce the problem to the triviality of a curve in a surface (see Sect. 5 for a description of the homotopy equivalence). In order to do so despite the compression, we compute in Sect. 4 a *retriangulation* of the surface, so that the curves in Δ are only polynomially long in the new triangulation. This is done by tracing the normal curve and building the *street complex* which was specifically designed by Erickson and Nayyeri to handle normal curves on surfaces in polynomial time. We then track how the street complex interacts with our input curve γ. This operation comes at a cost: even if the input curve γ was not compressed, it may become exponentially long after the retriangulation. Since it may be not simple, we rely on straight-line programs instead of normal curves to encode it. At this stage, we note that it might as well have been a compressed straight-line program from the start.

Finally, we want to test the triviality of a compressed curve in a wedge of surfaces. The fundamental group of this wedge is a free product of fundamental groups of surfaces, and thus the crux of the problem is to solve it in a single surface. While there are known techniques to test the contractibility of a curve on a surface, based on local simplification rules [16, 30], the compressed nature of our curves is once again a stumbling block here, as we need to be careful to never apply an exponential number of such simplification rules. The solution sketched in [40, App. A] to that issue is to introduce a family of *well-tempered paths*, which are carefully designed to not necessitate such exponential simplifications. We use a different approach: we rely on the standard tools developed for the non-compressed case, namely quad systems and turn sequences, and detect exponential simplification and realize them all at once in polynomial time. This relies on recent insights on the structure of these quad systems, [11], [31, Sect. 4.3].

2 Background

We begin by recalling some key definitions and results, although we assume that the reader is familiar with basic algebraic topology such as homotopy and fundamental groups; we refer to [21] or [42] for more detailed definitions.

2.1 3-Dimensional Manifolds

A 3-manifold is a topological space that is locally homeomorphic to a 3-dimensional ball or a 3-dimensional half-ball. The points of the latter type form the *boundary* of the 3-manifold, which is always a surface. In this paper, 3-manifolds are always assumed to be triangulated, and we use common and a less restrictive definition than simplicial complexes: a *triangulation* of a 3-manifold M is a collection of n abstract tetrahedra, along with a collection of gluing pairs which identify some of the $4n$ boundary triangles in pairs. In particular, we allow two faces of the same tetrahedron

[3] A *wedge* of a family of topological spaces is the space obtained after attaching them all to a single common point. Actually, there are also circle summands here—we do not mention them in this outline to keep the discussion light.

to be identified. If we add the further restriction that the neighborhood of each vertex is a 3-ball or a half 3-ball and no edge gets glued to itself with the opposite orientation, then the resulting quotient space is always a 3-manifold [37]. We use $|M|$ to denote the number of tetrahedra of a triangulated 3-manifold M.

We briefly recall normal surface theory, which will be used in Sect. 3, and refer the reader to [19] and [6] for a more detailed overview and background of the topic. A *normal isotopy* is an isotopy of a triangulated 3-manifold that leaves each cell of the triangulation invariant. A *normal surface* in a triangulation is a properly embedded surface transverse to the triangulation; such a surface decomposes into a disjoint collection of *normal disks*, each embedded in a single tetrahedron. Each disk inside a tetrahedra is either a triangle, which separates one vertex in a tetrahedron from the others, or a quadrilateral which separates two vertices from the other two. In each tetrahedron, there are (up to normal isotopy) four types of such triangles, and three types of quadrilaterals. The normal surface can be reconstructed using only the number of these triangles and quadrilaterals in each tetrahedron, as long the collection of disks satisfies a family of equations called *matching equations* and has at most one quadrilateral type in each tetrahedron. These numbers are called the *normal coordinates*, and since n bits encode numbers up to 2^n, these normal coordinates naturally have exponential compression: normal coordinates of polynomial bit-size encode surfaces containing possibly exponentially many normal disks. Nevertheless, one can compute the Euler characteristic of a normal surface in polynomial time in the bit-size encoding, since it is a linear form [19, formula (22)]. Finally, *crushing* a triangulation T along a normal surface roughly consists, for each tetrahedron containing at least one quadrilateral, in removing that quadrilateral and gluing its neighbours pairwise along the two pair of edges not intersected by the quadrilateral, as in Fig. 1. Then one tears apart in multiple components the triangulation at the places where it is not a 3-manifold. This can be done in polynomial time, and surprisingly, this very destructive process has a well-controlled topological meaning when the normal surface is a sphere [6, Cor. 5].

2.2 Curve and Surface Representations

All the surfaces in this article are closed, i.e., compact and without boundary. We shall often represent a surface by a graph cellularly embedded in that surface. The homeomorphism class of this embedding is encoded as a *combinatorial surface* thanks to a rotation system over the edges of the graph [36]: intuitively, this encodes the circular permutation of the edges around each vertex. Equivalently, one can define a combinatorial surface as a gluing of polygons whose sides are pairwise identified. When all the polygons are triangles, we speak of a *triangulation*, or a *triangulated surface*. Note that we allow a triangle to have two sides identified after the gluing or two triangles to share two vertices but no edge, etc.

The *complexity* of a surface is the number of cells (vertices, edges and faces) of the complex induced by the graph embedding. A curve on such a surface can be represented or encoded in a variety of ways. For instance, any curve is homotopic to a curve defined by a walk on the 1-skeleton of the complex. The complexity of the curve is the number of edges in the walk counted with repetition. One can also consider curves in general position avoiding the vertices of the graph and cutting its edges transversely as in the next section, in which case we say the complexity of a curve is its number of intersection points with the graph.

We next outline two different notions of compressed curve representations. To repeat the footnote of warning in the introduction: normal coordinates and straight line programs are two different methods for compressed representations, both of which we use in this paper. We keep the name *compressed* for straight line programs, while a curve or a multicurve encoded with normal coordinates will be simply called a *normal (multi-)curve*.

2.2.1 Normal Curves

Let S be a triangulated surface. A *multi-curve* is a disjoint family of curves embedded on S. A multi-curve c is *normal* if it is in general position with respect to the triangulation of S and if every maximal arc of c in a triangle has its endpoints on distinct sides. Moreover, no component of a normal curve is contained in the interior of a single triangle. A normal curve may have three types of arcs in a triangle, one for each pair of sides. Counting the number of arcs of c of each type in each of the t triangles of the triangulation, we get $3t$ numbers called the *normal coordinates* of c. As for triangulated 3-manifolds, a *normal isotopy* of S is an isotopy that leaves each cell of the triangulation invariant. Up to a normal isotopy, c is completely determined by its normal coordinates. It is thus possible to encode a multi-curve with exponentially many arcs by storing its normal coordinates, which then have polynomial bit-complexity. A normal multi-curve is *reduced* if no two of its components are normally isotopic.

2.2.2 Straight-Line Programs

Another method of compressing curves is to think of a curve as a word over an alphabet. In this encoding, a letter of the alphabet corresponds to a directed edge on the surface so that a word corresponds to a juxtaposition of edges. Therefore, we can consider curves as abstract words and represent them using compressed representations of abstract words. The length of a word w, denoted by $|w|$, is the number of letters in it. We recall that the Kleene closure of an alphabet $\mathcal{A}$ is the set $\mathcal{A}^*$ of finite words on $\mathcal{A}$, including the empty word.

A *straight-line program* is a four-tuple $\mathbb{A} = \langle \mathcal{L}, \mathcal{A}, A_n, \mathcal{P} \rangle$ where:

- $\mathcal{L}$ is a finite alphabet of *terminal* characters,
- $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$ is a disjoint alphabet of *non-terminal* characters,
- $A_n \in \mathcal{A}$ is the *root*, and
- $\mathcal{P} = \{A_i \to W_i\}$ is a collection of *production rules*, one for every non-terminal, where W_i is a word in $(\mathcal{L} \cup \mathcal{A})^*$ such that every non-terminal A_j appearing in W_i has index $j < i$.

2.2.3 Example

Consider a graph with two nodes a and b and two oriented edges e and f from a to b. As an example let $\mathcal{L} = \{e, f, \bar{e}, \bar{f}\}$ consist of four oriented edges, where the bar denotes the edge with inverse orientation. Let $\mathcal{A} = \{A_1, \dots, A_n\}$ be the set of non-terminal letters. Intuitively, these are variables that represent sub-words of a word, and each one can be thought of as being a root of a sub-program. Let the rules of our program be $A_i \rightarrow A_{i-1}A_{i-1}$ for $i \geq 2$ and $A_1 \rightarrow f\bar{e}$. Then the root non-terminal represents the word (or the oriented closed curve) $(f\bar{e})^{2^{n-1}}$ and the non-terminal A^{n-j} represents the sub-word $(f\bar{e})^{2^{n-j-1}}$. Note that not all words over the alphabet of edges correspond to a connected curve.

To compute the word represented by a non-terminal A_i in a straight-line program, we need only replace A_i using the production rule $A_i \rightarrow W_i$, and then recursively replace the non-terminals in W_i until there is no non-terminal. The recursion tree of this procedure is called the *production tree* for the non-terminal.

A straight-line program is in *Chomsky normal form* if every production rule either has two non-terminals or a single terminal on the right. Every straight-line program can be put in polynomial time into Chomsky normal form by adding intermediate non-terminals, and thus we will always assume that a straight-line program is in Chomsky normal form. We denote by $w(A)$ the word encoded by a non-terminal character A, or sometimes simply by A when there is no confusion. Its length is denoted by $|A|$. We will always use a terminal alphabet $\mathcal{L}$ equipped with an involution $a \mapsto \bar{a}$. The *reversal* of a word w is the word obtained by changing its letter a by its reverse $\bar{a}$ and reversing the order of the letters.

Composition systems are straight-line programs of a more advanced type, where productions of the form $P \rightarrow A[i:j]B[k:l]$ are allowed, and the meaning is that $A[i:j]$ represents the subword comprising the letters from index i up to $j-1$ (both included, starting the indexing at 0) of the word that A encodes. We take the convention that negative indices count from the end of the word and that $A[i:]$, $A[i]$, and $A[:j]$ are shorthand respectively for $A[i:|w|]$, $A[i:i+1]$ and $A[0:j]$, where w denotes the word represented by A. We have for example $A[-k:] = A[|w|-k:|w|]$ and $A[:-k] = A[0:|w|-k]$. While composition systems might seem more powerful than straight-line programs, a theorem of Hagenah [17, Chap. 8] (cf. [40, Sect. 7]) says that given any composition system, one can compute in polynomial time an equivalent straight-line program. Henceforth, we will slightly abuse language and freely use the $[\,\cdot:\cdot\,]$ construct in our straight-line programs. We also identify a straight-line program with its root, writing for instance $\mathbb{A}[i]$ for $A_n[i]$.

The following theorem summarizes the algorithms on straight-line programs that we will rely on, see [40, Sects. 2 and 3] or [33].

Lemma 2.1 *Let $\mathbb{A}$ and $\mathbb{X}$ be two straight-line programs, i be an integer and e be a letter in the terminal alphabet of $\mathbb{A}$. Then one can, in polynomial-time,*

– *compute the length of $\mathbb{A}$,*
– *output the letter $\mathbb{A}[i]$,*
– *find the greatest j so that $\mathbb{A}[j] = e$,*

– *compute a straight-line program* $\overline{\mathbb{A}}$ *for the reversal word* $\overline{w(\mathbb{A})}$,
– *decide whether* $w(\mathbb{A}) = w(\mathbb{X})$,
– *find the biggest* k *such that* $\mathbb{A}[:k] = \mathbb{X}[:k]$.

Any closed curve in the 1-skeleton of a given complex K can be encoded by a straight-line program whose terminal alphabet is the set of directed edges of K. A trivial such encoding, not in Chomsky normal form, is given by a single production rule mapping the root to the curve itself. We call such an encoding a *compressed curve* or *walk*. Simple operations on K can seamlessly be done while updating a compressed curve appropriately to obtain a homotopic compressed curve in the modified complex. For example, contracting an edge e of K simply boils down to adding a production rule $e \to \varepsilon$, where ε is the empty word. Likewise, deleting an edge e bounding a face $\bar{e}p$, where p is the complementary path in that face, amounts to replacing each occurrence of e by a p via a new production rule $e \to p$, turned into Chomsky normal form. When no encoding is specified for a given closed (multi-)curve, then it is simply encoded as a (multi-)walk on the one-skeleton of the underlying surface or complex.

3 From Contractibility to Normal Subgroup Membership

The following proposition reduces the contractibility problem for closed curves on the boundary of an orientable 3-manifold M to the normal subgroup membership problem for a boundary surface of M, where the multi-curve Δ is a reduced normal curve with polynomial complexity. We note that this reduction is the only part that is NP in our algorithm, the rest of our algorithms run in deterministic polynomial time.

Proposition 3.1 *Let* M *be an orientable triangulated* 3*-manifold with boundary. There exists a reduced normal multi-curve* Δ *on* ∂M *with normal coordinates of linear bit-size* (*with respect to* $|M|$)*, so that for any curve* γ *in a surface component* S *on* ∂M*,* γ *is contractible in* M *if and only if* γ *is contained in the normal subgroup* N *of* $\pi_1(S)$ *generated by the curves in* $\Delta \cap S$.

Further, given M*,* Δ*, and a cubic-sized certificate* (*in* $|M|$)*, there is a polynomial-time algorithm to verify that* Δ *has the property that all curves in* N *are contractible in* M.

Proof A 3-manifold M is *irreducible* if every properly embedded 2-sphere in M bounds a 3-ball. Our first step is to transform M so that it is irreducible. A *reducing sphere* is a properly embedded 2-sphere that does not bound a 3-ball. If there exists a reducing sphere, there exists one that is normal, with normal coordinates of linear bit-size [34, Prop. 3.3.24 and Thm. 4.1.12]. Using the techniques of Jaco and Rubinstein [24] and Burton [6] one can *crush* [6, 24] along such a normal surface to obtain a new triangulated manifold M' with strictly less tetrahedra. Note that since our 3-manifold is orientable, there are no 2-sided projective planes, and thus the crushing is well-defined.

We claim that M' has boundary identical to that of M. Indeed, a normal sphere (and, more generally, any closed normal surface) does not have quadrilateral faces in the tetrahedra with at least one face on the boundary, and thus those are unharmed by

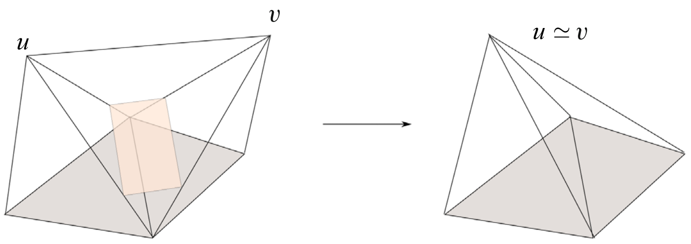

Fig. 1 The boundary is shaded in gray. Tetrahedra with a triangle on the boundary do not contain quadrilateral normal disks, but tetrahedra with one edge on the boundary might contain one type of those. Crushing along this quadrilateral glues the adjacent tetrahedra together and thus preserves the boundary

the crushing procedure. Furthermore, the tetrahedra with one edge on the boundary can only carry a single quadrilateral type, whose crushing along glues together the adjacent tetrahedra with faces on the boundary, see Fig. 1. Since the boundary stays the same, crushing along a reducing sphere only undoes connected sums and removes connected summands without boundary [6, Cor. 5]. Since connected sums induce free products of fundamental groups, a curve on $\partial M'$ is contractible in M' if and only if it is contractible in M. We iterate this procedure on M' until we get a 3-manifold M^i that is irreducible.

A *compression disk* for the boundary of a 3-manifold M is a topological disk D that is properly embedded and such that ∂D is an essential curve in ∂M. Classic theorems of normal surfaces (see for example [25, Sect. 6]) show that there exists a complete family $\mathcal{D}$ of linearly many, that is $O(|M^i|)$, disjoint compression disks for ∂M^i which are *vertex normal surfaces*. Moreover, we can assume the set of disks $\mathcal{D}$ is *minimal* in the sense that no subset of it is a complete family of compression disks.

Without delving into the details of normal surface theory, we just summarize the salient features of these vertex normal disks for our purpose: the elements of $\mathcal{D}$ are disks of which the boundaries are normal curves on ∂M^i. Each of these boundaries might be exponentially long, but their coordinates, considered as a single multi-curve Δ, fit in a quadratic number of bits (see for example [19]). Furthermore, this family is complete, i.e., if we denote by C the 3-manifold which is a regular neighborhood of $\partial M \cup \mathcal{D}$ (C is an example of a compression body), then $\partial C \setminus \partial M$ is incompressible in C as well as in $\overline{M \setminus C}$.

Let γ be a closed curve on $\partial M = \partial M^i$, and let S be the surface component of ∂M that contains γ. We denote by N the normal subgroup of $\pi_1(S)$ generated by the curves of Δ in[4] S (after attaching all the curves at a common base point). We claim that γ is contractible in M, hence in M^i, if and only if it belongs to N. If γ belongs to N, then it is clearly contractible, because all generators of N are so. Now let us assume that γ is contractible, but it does not belong to N. Then according to the loop theorem [22, Thm. 4.2], there exists a properly embedded disk D in M^i so that ∂D belong to $\pi_1(S) \setminus N$. We pick such a properly embedded disk D so that the number of connected components of $D \cap \partial C$ distinct from ∂D is minimal. This number is non-zero as otherwise D is contained in C and thus ∂D is contractible in

[4] Their normal coordinates are given by the restriction on S of those of Δ.

C whose fundamental group is isomorphic to $\pi_1(S)/N$, thus belongs to N, which is a contradiction. Therefore there is an innermost subdisk $D' \subseteq D$ so that $\partial D' \subseteq \partial C$. This disk D' is either contained in C or in $\overline{M \setminus C}$, in both cases this contradicts the completeness of $\mathcal{D}$; indeed $\partial D'$ cannot bound a disk in ∂C by the minimal choice of D, hence D' must be a compression disk.

For the verification part, the certificate consists of $\mathcal{D}$, as well as all the reducing normal spheres S^j and the intermediate manifolds M^j. Note that since each crushing strictly reduces the size of the triangulation, there are linearly many of those. One can verify in polynomial time that each S^j is a connected sphere (see [1, Sect. 6]), and that crushing along it yields the next manifold: despite the sphere having potentially exponentially many normal disks, the crushing procedure only looks at the existence of quadrilaterals in tetrahedra, which is done in polynomial time. Finally, one checks that each connected component of $\mathcal{D}$ is a disk. By the previous arguments, this ensures the required property. This certificate has cubic size, since there are a linear number of disks and spheres, each of quadratic bit-size (see the bounds on the coordinates of vertex normal surfaces in [19, Sect. 6]).

We finally note that the multi-curve must be reduced. Otherwise, assume there are two components that are isotopic on the boundary. The two normal disks bounding the curves and the image of the isotopy define a 2-sphere which has to bound a ball, thus contradicting the fact that Δ is minimal.

Note that we did not strive to optimize the size of the certificate in this proposition, since it does not matter for our application, so further improvements may be possible. In particular, using enumeration techniques for vertex surfaces [19, Sect. 6], we suspect it can be shown to be quadratic rather than cubic.

Proposition 3.2 *The problem of contractibility of a compressed curve on the boundary of an arbitrary* 3*-manifold* M *reduces, in polynomial time, to the problem of contractibility of a compressed curve on the boundary of an orientable* 3*-manifold.*

Proof Let T denote the triangulation of M and let c be any closed curve lying on ∂M. We can construct a triangulation of the orientable double cover $\tilde{M}$ of M from T in linear time. For example one can take two copies of each tetrahedron, one for each orientation, which induces orientations of their faces. Then one identifies pairs of faces whenever they are identified by T and the induced orientations of the faces are reversed. If c lifts to a closed curve $\tilde{c}$, then c is contractible in M if and only if any lift $\tilde{c}$ is contractible in $\tilde{M}$. If c does not lift to a closed curve then c is not contractible in M.

To finish the proof we need to present a polynomial-time algorithm that given a compressed curve γ, computes a compressed curve γ' in the double covering such that $p(\gamma') = \gamma$ where p is the covering map. We can assume that the compressed word γ is given in Chomsky normal form. We lift the sub-paths of γ in the two possible ways starting from the leaf production rules. Therefore, when we have a rule of the form $A \to A_i A_j$ we already have computed two preimages $\tilde{A}_i$ and $\tilde{A}'_i$, and, similarly, $\tilde{A}_j$ and $\tilde{A}'_j$. We compute a preimage of A by looking at the last vertex of $\tilde{A}_i$, if this vertex equals the first vertex of $\tilde{A}_j$ we set $\tilde{A} = \tilde{A}_i\tilde{A}_j$. Otherwise the vertex has to be the same as the first vertex of $\tilde{A}'_j$, we then set $\tilde{A} = \tilde{A}_i \tilde{A}'_j$. We define $\tilde{A}'$ analogously by starting from $\tilde{A}'_i$. At the end, each of the terminals corresponding to the root terminal defines a preimage of the curve γ in the double cover. □

Relying on Proposition 3.2, for the rest of the paper we restrict our attention to orientable surfaces and 3-manifolds.

Hardness of the general normal subgroup membership problem If the curves Δ in the normal subgroup membership problem are not disjoint then this problem is much harder, and possibly undecidable, even if they have few edges. We now elaborate on the details. To this end, recall that the disjoint normal subgroup membership problem is defined as follows. Given a family of disjoint normal closed curves Δ on a triangulated surface S, and a curve γ on S, decide if γ lies in the normal subgroup of the fundamental group of S determined by the curves of Δ (appropriately connected to the basepoint).

If we can divide Δ into two subsets, each of which contains only disjoint curves, then it is not hard to see that the problem is equivalent to contractibility in 3-manifolds. Take an arbitrary triangulated 3-manifold M and thicken its 1-skeleton inside M. Define S to be the boundary of the thickening of the 1-skeleton and let Δ be the "meridians" of the edges, and the intersections of the triangles with S. Given a closed curve γ whose contractibility is to be decided, isotope γ in M so that it lies in S. Then by gluing disks to Δ we obtain a space which has the same fundamental group as M. Therefore, the contractibility of curves in the interior of a 3-manifold reduces to the normal subgroup membership problem with non-disjoint curves. On the other hand, if we can partition Δ into two subsets Δ_1 and Δ_2, each of which contains only disjoint curves, then the normal subgroup membership problem reduces to contractibility of a curve in the interior of a 3-manifold by gluing disks to Δ_1 and Δ_2 on opposite sides of S, and then thickening.

If the curves Δ are arbitrary then the problem is undecidable. We can perform the same reduction as above but using a 4-dimensional manifold instead. It is known that the contractibility problem for 4-manifolds is undecidable since they can have any finitely presented group as fundamental group [42, Exer. 8.1.8.1].

4 Retriangulation

Throughout this section Δ denotes a reduced normal multi-curve on a triangulated 2-manifold. In what follows, we present a way to compute a new triangulation of polynomial complexity, in which the curves of Δ lie in the 1-skeleton and are only polynomially long. Techniques exist (see e.g., [5] or [15]) to retriangulate in this way. However, we also must track our compressed word γ in this new triangulation without increasing its (compressed) complexity during the retriangulation process. We remark, as mentioned in the introduction, that even if we start with a curve γ which has polynomially many edges, the image of γ in the new triangulation may have exponentially many edges. In fact, if some normal coordinate of Δ encodes an exponential number in the size of the input triangulated 2-manifold, then γ will cross the new triangulation exponentially many times as soon as γ traverses one of the two edges corresponding to this normal coordinate in the input triangulation.

Proposition 4.1 *Let a simplicial complex K triangulating a surface S, a reduced normal multi-curve Δ and a compressed closed walk γ on the 1-skeleton of K be given as input. The size of the input is the summation of the complexities of the curves γ, Δ and complexity of K. There is an algorithm that in polynomial time computes a new simplicial complex K' triangulating the same surface S, a multi-curve Δ', and a walk γ' on the 1-skeleton of K' given as a composition system, such that:*

- *there is a homeomorphism $f\colon \|K\| \to \|K'\|$, such that the images of Δ under f coincides with Δ',*
- *γ' is homotopic to $f(\gamma)$,*
- *the multi-curve Δ' is an embedded multi-curve on the 1-skeleton of K'.*

Consequently, the homotopy class of γ' belongs to the normal subgroup generated by the components of Δ' in $\|K'\|$ if and only if the homotopy class of γ belongs to the normal subgroup generated by the components of Δ in $\|K\|$.

Note that in this proposition, the complexities of the inputs γ and Δ is the size of their compressed and normal encodings.

Proof A re-triangulation satisfying the conditions on Δ and Δ' and disregarding γ can be constructed by at least two explicit polynomial-time algorithms in the literature, one by Erickson and Nayyeri [15] and the second by Bell [5]. We work with the former, basing our re-triangulation on their *street complex*, and introduce additional structure to track γ.

We briefly recall the structure of street complex here. The overlay of Δ with K cuts the triangles of K into smaller pieces. We abuse language and call *quadrilaterals* the pieces bounded by exactly two sub-edges of K, though they can actually be degenerate triangles. Merging the quadrilaterals whenever they share an edge segment of K results in the *street complex*. The maximal sequences of merged quadrilaterals are called *streets*, and the remaining faces are *junctions*. Quite surprisingly, the size of the street complex is bounded by a linear function of the complexity of K [15, Lem. 2.3], independent of Δ. Note that the curves in Δ appear in the 1-skeleton of this complex at the street boundaries. The complex K' in the proposition is essentially this street complex, further triangulated, and Δ' is the image of Δ in K'.

We model a street as a rectangle whose boundary is a simple cycle consisting of some vertices and edges. A generic street has two edges on the boundary that are segments of edges of K, these are denoted by the ϵ_i in Fig. 2(a). The boundary minus ϵ_1 and ϵ_2 is divided into two paths that we call the *sides* of the street. In general, our model of a street is mapped to a street from [15] so that some pairs of the edges might identify with each other. Any pair of identified edges has one edge in each side. We draw these edges using a thick line. Fig. 2(b) depicts a street model that is mapped to a spiral, shown in (c).

The input compressed curve γ is a walk on the 1-skeleton of K. We may assume that the straight-line program encoding γ is in Chomsky normal form (cf. Sect. 2). The production rules with a terminal at the right-hand side now are of the form $A \to e$

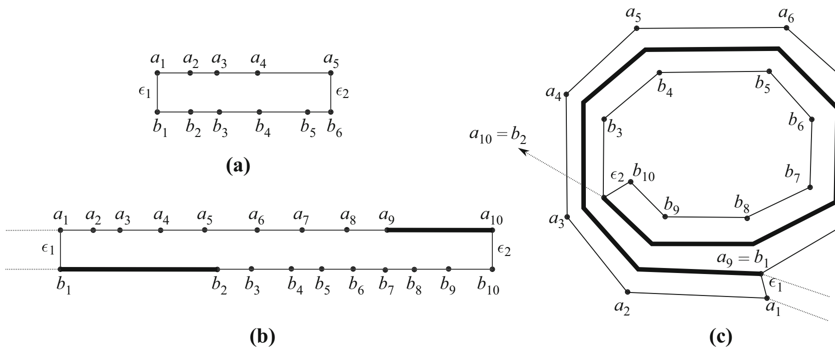
a_1 a_2 a_3 a_4 a_5
ϵ_1
ϵ_2
b_1 b_2 b_3 b_4 b_5 b_6
(a)
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10}
(b)
$a_{10} = b_2$
$a_9 = b_1$
(c)

Fig. 2 **a** A model for a street without a spiral, **b** a model for the spiral, where two edges on the boundary (shown in bold) are identified, **c** the spiral redrawn as embedded on the surface

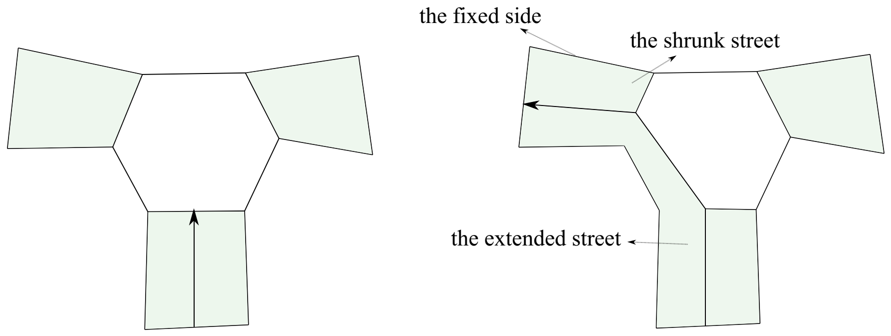

Fig. 3 A left turn in the street complex; the arrows depict segments of the edge *e*

where *e* is an oriented edge of the 1-skeleton of K. We trace the multi-curve $\varDelta$ using the algorithm of [15, Sect. 4.2], but at the same time define a composition system $\mathbb{E}$ encoding the intersections of the edge *e* with the street complex for $\varDelta$. We update the composition system $\mathbb{E}$ after each step of the tracing algorithm, as described below.

The basic *step* of the tracing algorithm is to extend a street across a junction or a fork (we extend across a fork when the curve we are tracing intersects an edge containing an endpoint of the trace), and then across a street. The extension across a junction and a street is depicted in Fig. 3. This figure depicts a left turn across a junction. When a street is extended through a second street, the latter street is shrunk. When a street is shrunk, one side of the street remains unaltered, we call this side the *fixed side* of the street. The boundary of the street which is extended gains some new vertices and edges of the street complex, while the boundary of the street which is shrunk becomes simpler. The effect of this step on the streets is depicted in Fig. 4.

We next recall that a *phase* of the tracing algorithm of [15] consists of a maximal sequence of left turns or of right turns. When we enter a street for the second time, from the same boundary edge, during the same phase, we have discovered a *spiral*. In [15] it is shown that the spirals can be processed at once in $O(m)$ time, where m

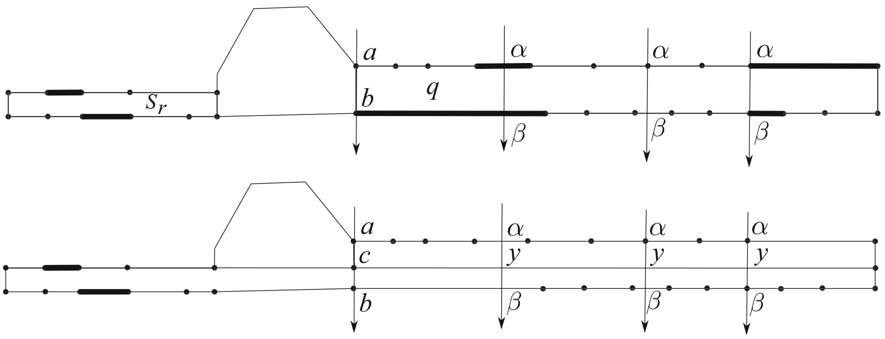

Fig. 4 The effect of a right turn on the model streets, the arrows depict parts of the edge e. Top: before the extension. Bottom: after the extension of the street s_r along the street q

is the number of edges of K. The presence of these spirals in a street causes potential identifications; we call the edge which is the result of an identification the *long edge* of the spiral (Fig. 2). Observe that when another street extends over a street that contains a spiral, the latter street turns into an ordinary street without identifications. Since we are only interested in the case where Δ is reduced, the complexity of the street complex is always bounded by a linear function of the complexity of K [15, Lem. 2.3].

The terminals of our composition system $\mathbb{E}$ are of three types: i) (s, a, x) or (s, x, a), where s is a street, a is a vertex on s, and x is an edge of the street on the opposite side from a, ii) (s, x, y) where x and y are edges on opposite sides on the model of street s, and iii) (s, a, b) where a and b are vertices of the model street for s lying on opposite sides. These three types of terminals represent sub-arcs of e that lie inside the street, where the sub-arcs start at the second parameter and end at the third parameter. We also define types for non-terminals, which represent longer parts of an edge analogously. We use vertices and edges of the model street rather than those of the street itself. This allows us to distinguish between two arcs in the street complex, which have different directions, but connect the image of two edges or vertices that are identified in the street complex.

When traversing an edge $e = uv$ of γ in the overlay of the street complex and the original complex K, we see a sequence of these terminals which we call the *overlay sequence* of e. Our first goal is to construct a composition system for the overlay sequence.

The algorithm that traces the curves in Δ to build the street complex starts with the triangulation K, where each edge is a (trivial) street and there is one fork where we start the tracing. At the beginning, we introduce for the edge $e = uv$ of γ a non-terminal $A(s_e, u, v)$, where s_e is the degenerate street corresponding to the edge e. We also introduce the rule $E \to A(s_e, u, v)$. This finishes the modifications for the starting street complex.

We now explain the rules that are added to the system to account for an ordinary street extension. Assume that we turn right at a junction. Referring to the arrows in Fig. 4, there are four cases to consider. These cases correspond to different possibilities for the start and endpoint of a segment of e that crosses the street q which is going to be shrunk. This segment is represented by a non-terminal which will be the left-hand side of the new rules.

Let a and b be endpoints of a side of the street q that lies on an edge, and let s_r be the right active street, as in Fig. 4. Note that each street is a rectangle, so we can talk about its sides. If the non-terminal $A(q,a,b)$ exists, we introduce the rule $A(q,a,b) \to A(q,a,c)\,A(s_r,c,b)$ where c is a vertex as in the figure. If the street coordinates of Δ in (new) q and/or in s_r is 0, we also introduce the rule $A(q,a,c) \to (q,a,c)$ and/or $A(s_r,c,b) \to (s_r,c,b)$ and the corresponding terminals. Similarly, if the non-terminal $A(q,b,a)$ exists we introduce the rule $A(q,b,a) \to A(s_r,b,c)A(q,c,a)$. We introduce also the suitable terminals in case any of the streets have coordinates zero for Δ. Recall that during the tracing algorithm of [15], the number of arcs of Δ inside each street is maintained as the street coordinates.

For any pair (α,β) where α and β are a vertex or an edge of the street q, we search if there exists already a non-terminal of the form $A(q,\alpha,\beta)$. If the non-terminal $A(q,\alpha,\beta)$ exists, and α lies on the fixed side of q, we introduce the rule $A(q,\alpha,\beta) \to A(q,\alpha,y)\,A(s_r,y,\beta)$ where y is the edge of (new) street q as in the bottom of Fig. 4. If the street coordinates of Δ in q and/or in s_r is 0, we also introduce the rule $A(q,\alpha,y) \to (q,\alpha,y)$ and/or $A(s_r,y,\beta) \to (s_r,y,\beta)$ and the corresponding terminals. Similarly, if the non-terminal $A(q,\alpha,\beta)$ exists where β is on the fixed side of the street q we introduce the rule $A(q,\alpha,\beta) \to A(s_r,\alpha,y)\,A(q,y,\beta)$. We introduce also the suitable terminals in case any of the streets have coordinates zero for Δ. This finishes the processing of non-terminals.

The modifications necessary for extension across a fork are totally analogous to a junction. In this case again a street is shrunk and another is extended, see [15, Fig. 7]. We do not repeat the case analysis.

We next consider a spiral. Let d be the depth of the spiral, l the length of the spiral, and m be the number of distinct streets traversed by the spiral. See [15] for more detailed description of these parameters and how to compute the spiral. Let λ be the long edge of the spiral, and λ_1, λ_2 be its preimages in the model. After we enter a spiral and before we reach the “final turn", the new side of the shrunk street is always λ and the extended street has the edge λ on both sides. See Fig. 5. Let, in the system at the time we discover the spiral, $A(s_i,x,y)$ be the non-terminal where s_i is a street traversed by the spiral, and x and y be edges on its two sides such that x lies on the fixed side of s_i. After we trace the whole spiral, we need to add the rule $A(s_i,x,y) \to A(s_i,x,z)\,A(s_a,z,\lambda_*)\,A(s_a,\lambda_*,\lambda_*)^{d'}!A(s_a,\lambda_*,y)$, where s_a is the active extended street. In place of the star we place indices 1, 2 as is appropriate. The number d' is either $d-1$ or $d-2$ depending on where the street lies around the spiral. For an example see Fig. 5. This figure depicts a clockwise spiral. In the highlighted street of the figure we have $d' = d-1$, and $d = 3$. If any street coordinate in the streets s_i becomes zero we introduce the rule $A(s_i,\lambda_*,x) \to (s_i,\lambda_*,x)$. If the active street has coordinates zero for Δ we add the rule $A(s_a,\lambda_*,\lambda_*) = (s_a,\lambda_*,\lambda_*)$. If instead of x, y is on the fixed side of the street s_i we do analogously. We perform these modifications for every existing non-terminal of the form $A(s_i,x,y)$. This finishes the modifications of the composition system for the non-terminals of type ii). Those of other two types are handled similarly.

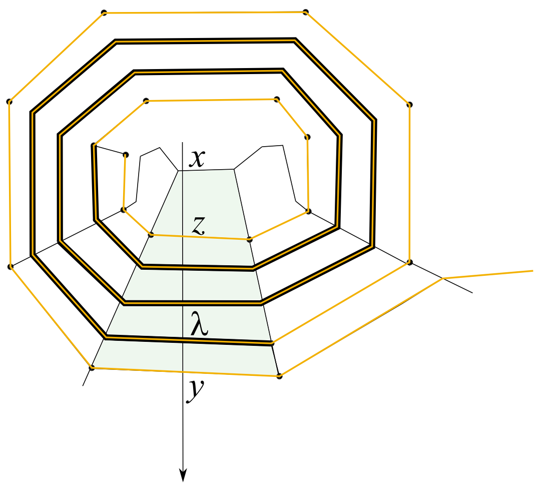

Fig. 5 Updating the intersections of e with a spiral, the orange curve is part of Δ

We have finished the construction of the composition system $\mathbb{E}$. Throughout the construction, we have added a number of non-terminals that is polynomial since the number of streets, vertices and edges is polynomial at each moment, and the number of steps in the tracing algorithms is also polynomial. Therefore the entire composition system is polynomial and can be computed in polynomial time. Note that since the curve Δ is reduced no street is an annulus. We replace each rule $A \to e$ of the walk γ by a rule $A \to E$ where E is the root of the composition system $\mathbb{E}$ encoding its overlay sequence with the street complex. To construct γ' we do as follows. For each edge of the the street complex, we choose one of the its endpoints. We then replace the terminals of the system $\mathbb{E}$ with a path on the boundary of a street between two vertices. This transforms the composition system into a composition system for the curve γ' which is freely homotopic to γ, and γ' is a walk on the 1-skeleton of the street complex.

To finish the proof, our final step is to turn the street complex into a triangulation. We simply need to triangulate streets and junctions by adding diagonals. We eventually apply at most two barycentric subdivisions to make sure that the result is a simplicial complex. We also modify the composition system for γ' to reflect the subdivision. Assume K' is the resulting simplicial complex. There exists a homeomorphism $f : \|K'\| \to \|K\|$. □

5 Capping Off

In this section, we show that the complex obtained by gluing disks on a family of disjoint curves on a surface is homotopy equivalent to a wedge of surfaces and circles. We provide a polynomial-time algorithm to compute the resulting wedge, while also tracking what happens to a compressed curve on the original surface.

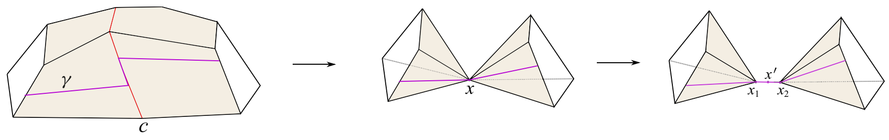

Fig. 6 Contraction of a component c of Δ (shown on the left) results in a new complex K' (shown in the middle). For clarity, we have not drawn all the edges

Proposition 5.1 *Let L be simplicial complex triangulating a surface S of genus ≥ 1, Δ be a family of disjoint embedded closed curves in the 1-skeleton of L and γ be a compressed closed walk on the 1-skeleton of L. The size of the input is the summation of the complexities of L, γ and the number of edges of Δ. We can compute in polynomial time a simplicial complex K which is a wedge of surfaces and circles, and a compressed walk w on the 1-skeleton of K, so that:*

- *K is homotopy equivalent to the complex obtained from L by gluing disks on each component of Δ,*
- *w is homotopic to a trivial walk in K if and only if γ belongs to the normal subgroup determined by the curves in Δ.*

Proof Note that the second property will be satisfied if, in the complex K satisfying the first property, we take for w the image of γ under a homotopy equivalence of L to K. We construct the simplicial complex K which is homotopy equivalent to the space obtained from L by gluing a disk to each component of Δ. This is done one disk at a time, and at each step we also maintain the image of the curve γ as a compressed word. Suppose Δ has m components. As a preprocessing step, we subdivide twice any edge of L that connects vertices on Δ but does not lie on Δ and also subdivide the incident triangles, and then update the compressed word.

Call a space X a *pinched surface* if there is a finite set of vertices P such that the closure of $X - P$ is a surface. We call P the set of *pinch points* and assume P is minimal, in the sense that no open neighborhood of a pinch point in X is homeomorphic to $\mathbb{R}^2$.

We start by computing a straight-line program in Chomsky normal form for γ. During the procedure that processes components of Δ, we maintain a complex K' and a compressed curve w' with the following invariants: (i) the current space $\|K'\|$ is a pinched surface, (ii) the current space $\|K'\|$ is homotopy equivalent to $\|L\|$ union caps (i.e., glued disks) over components of Δ that we have processed so far, (iii) the current curve w' is the image of the original curve γ under a homotopy equivalence and (iv) the (yet unprocessed) curves of Δ are sent by the homotopy equivalence to a set of disjoint simple curves Δ', at pairwise distance at least 2 on K'. By a slight abuse of notation we will identify Δ' and Δ in the following description.

Let c be a component of Δ and let K' be the current complex. We add a disk with boundary c; we then compress this disk into a single vertex x and add x to P. See Fig. 6. The effect on the word w' is as follows. A terminal $e = uv$ where uv is an edge in c is replaced with the empty word. A terminal $e = uv$, where v is in c but u is not, is replaced with $e' = ux$. If u is in c and v is not, the oriented edge $e = uv$ is replaced with $e' = xv$. It easily seen that these modifications can be accomplished in polynomial time. We note that each remaining component of Δ is an embedded curve in the 1-skeleton lying in a single component of $|K'| - P$. The invariants remain true, and the word w' is now a walk in the new complex K'.

After contracting all the components of Δ we have a complex K' which is a pinched surface. Because of the subdivision in the preprocessing step, the star of each pinch point x is the union of two disks $D_1 \cup D_2$ glued at their common center x. We further transform K' into a homotopy equivalent complex by replacing each such $D_1 \cup D_2$ with two disjoint disks connected through their centers by a path of two edges $x_1x'x_2$ as shown on Fig. 6. This homotopy equivalence simply expands x to $x_1x'x_2$. We then compute a tree in the 1-skeleton of the complex that spans the vertices x_1, x_2 for $x \in P$. We require that the number of x' vertices is minimal in this tree. We then contract every edge of the spanning tree. The result is a complex K which is a wedge of circles and surfaces, which can be turned into a simplicial complex using a constant number of subdivisions.

We explain the modifications needed to maintain the word w'. The first step is when we introduce the paths $x_1x'x_2$. Define $e_0 = x_1x'$, $e_1 = x'x_2$, and let $\bar{e_i}$ denote the edges with the opposite orientation. Observe that the link of the vertex x consists of two disjoint circles. We associate x_1 to one of these and x_2 to the other arbitrarily. If a terminal is an edge $e = ux$, where u is in the circle associated to x_2, we replace e with the path $e'\bar{e}_1$, where $e' = ux_2$. If u is in the circle associated to x_1, we replace e with the path $e'e_0$ where $e' = ux_1$. We do analogously for the edges of the form $e = xv$. These modifications define a curve which is the image of the homotopy equivalence which introduces the paths. There remains to update w' under contraction of a single edge. The result is the required curve w. ◻

6 Triviality for Compressed Words in Free Products of Surface Groups

In this section, we prove the following theorem:

Theorem 6.1 *Let K be a wedge of combinatorial surfaces and circles and w be a compressed walk on the 1-skeleton of K. Then one can compute in polynomial time a canonical form for the homotopy class of w. In particular, we can test in polynomial time whether w is trivial.*

We emphasize that in Theorem 6.1, we consider w as a walk with fixed endpoints, and thus we are considering based homotopies (as opposed to free homotopies).

We can assume that there are no spheres in the wedge of surfaces: since these are simply connected, removing them, and replacing their edges by empty letters in the word w does not change the homotopy class, or contractibility of w. Furthermore, tori require a slightly different treatment from the other surfaces. For the ease of exposition, we first explain and justify our algorithm assuming that there are no tori in the wedge, and at the end of the proof we will explain how to deal with them.

Our algorithm starts in a similar way as the known linear-time algorithms to test contractibility or homotopy of curves on surfaces [16, 30]: we first turn each surface in the wedge into a system of quads (Lemma 6.2), and then compute a canonical form for our curve w (Lemma 6.7). This canonical form is unique, and therefore the walk

w is homotopic to a trivial walk if and only if its reduced canonical form is the empty walk. However, due to the compression of the input, our techniques to reduce the curve w are more involved than those of the aforementioned references. Let $\mathbb{A}$ denote the straight-line program encoding w. First, we fix once and for all an orientation for all the surfaces in K, which gives precise meaning to the intuitive notions of turning right, clockwise, etc.

Our first step is to turn our surfaces into a *system of quads*. First, for each surface S we compute a spanning tree and contract it. Then, we remove edges until there is a single face, we add a new vertex inside the single face, add all the radial edges between this new vertex and the single vertex of S and remove all the previous edges. To make things unified, in each circle summand, each loop is subdivided into two edges. The resulting complex is called a *quad system*. We refer to [16, Sect. 3] for more details about the construction of quad systems.

Lemma 6.2 *Let K be a wedge of combinatorial surfaces and circles and let w be a compressed walk on the* 1*-skeleton K. In polynomial time we can turn K into a quad system K' and compute a compressed walk w' on* (*the edges of*) *K' that is homotopic to w.*

Proof The construction of the quad system follows directly from the definition. The word w is modified as follows: each terminal character representing an edge e (which got removed) is now a non-terminal character with a production rule $e \to e_1e_2$ where e_1e_2 is a two-edge path homotopic to e going through the new middle vertex. On surfaces, there can be two such edge-paths, we choose arbitrarily. The other production rules are unchanged. This operation can be done in polynomial time since there are polynomially many non-terminals and yields a compressed curve homotopic to w. □

In the next step of our algorithm, we encode words using *turn sequences*. Recall that a quad system consists of rectangles and has two vertices. The rectangles divide a small disk centered at a vertex into slices that we call *corners*. For any two directed edges e and e' on the same surface S, the turn $\tau(e, e')$ is the number of corners, counted with respect to our chosen orientation, between the end of e and the start of e' on S. That is, the number of corners we need to turn when we arrive at the vertex at the end of e and want to continue on e'. For any two edges e and e' forming a loop in a circle summand, we define $\tau(e, e')$ to be a new symbol λ, and for an edge e in a circle summand, we define $\tau(e, e^{-1})$ to be 0, which is in line with the case of surfaces. We need our encoding to adapt to jumps between different wedge summands, and we do it as follows: If e and e' are on different surfaces or circle summands, we define the turn $\tau(e, e')$ to be the pair ee', with the intended meaning that in this case the turn is a jump from e to e' on the new surface. Turn sequences are to be understood modulo the degree of the vertices, and following the literature, we use the notation $\bar{x} := -x$. The symbol λ satisfies $\bar{\lambda} = \lambda$. Henceforth, by a slight abuse of language, we count circle summands among the surfaces since they will be treated the same way. The turn sequence of our word is the concatenation of all the turn sequences of consecutive edges. However, we lose information by encoding with turn sequences. First, a turn sequence does not specify a starting edge. Second, even if we know the starting edge, it is not immediately clear how to compute an arbitrary intermediate edge along the

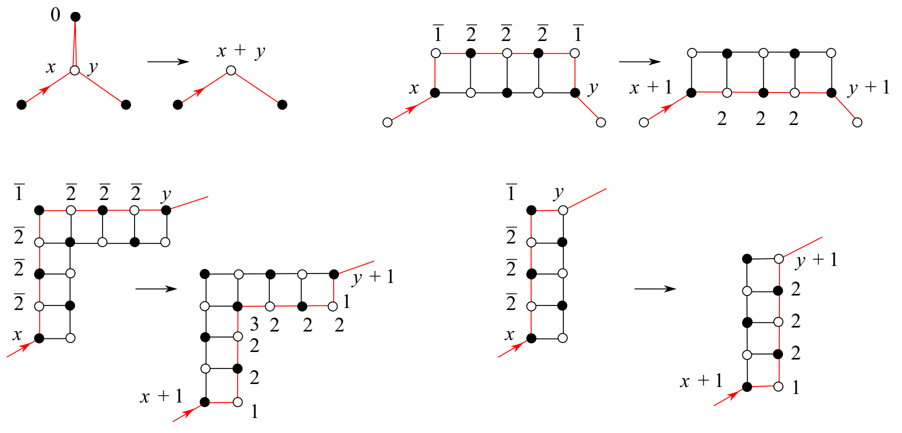

Fig. 7 The reductions to eliminate a spur (top left) and a bracket (top right), and the shifting moves to remove a $\bar{1}$ (bottom row)

path efficiently since we encode exponential length paths. The following lemma shows how to do that in polynomial time.

Lemma 6.3 *Let $\mathbb{T}$ be a compressed turn sequence and k be an integer. Given a starting edge e_{initial}, we can compute in polynomial time the edge of K we are on just after starting at e_{initial} and following the turn sequence up to $\mathbb{T}[k]$ included.*

Proof We first compute a straight-line program for $\mathbb{T}[: k+1]$, then use Lemma 2.1 to compute the last letter of this word that corresponds to an edge, hence to a jump. We denote it by e_{last}, and if there is none we use e_{initial} instead. We denote by i the index where it appears, which is 0 if there is no such edge. Then we look at the straight-line program computing $\mathbb{T}[i : k+1]$, which thus stays on a single surface. We inductively go up the production tree, computing, for each directed edge e of the surface and for each production rule at which directed edge we arrive if we start at edge e. This is straightforward at the leaves of the tree, and for a directed edge e and a production rule of $\mathbb{A}$ of the form $A_i \to A_j A_k$, we can look up where we arrive after starting at e and following A_j, denote the resulting edge by e', and then look up where we arrive after starting at e' and following A_j. Finally, we obtain our solution by looking up where we are after starting at e_{initial} and following $\mathbb{T}[i : k+1]$. □

The point of turn sequences is that they make easier the local simplifications to a contractible curve. Following Erickson and Whittlesey [16, Sect. 4], we use the following simplification rules, see Fig. 7, which are homotopies.

- A *spur* in a turn sequence w is a 0 turn. Removing a spur is applying the rule

$$x0y \to x + y.$$

- A *bracket* in a turn sequence w is a subword of the form $12\dots21$ or $\bar{1}\bar{2}\dots\bar{2}\bar{1}$. Flattening a bracket is applying the rules

$$x12\dots21y \to (x-1)\bar{2}\dots\bar{2}(y-1) \quad \text{or} \quad x\bar{1}\bar{2}\dots\bar{2}\bar{1}y \to (x+1)2\dots2(y+1).$$

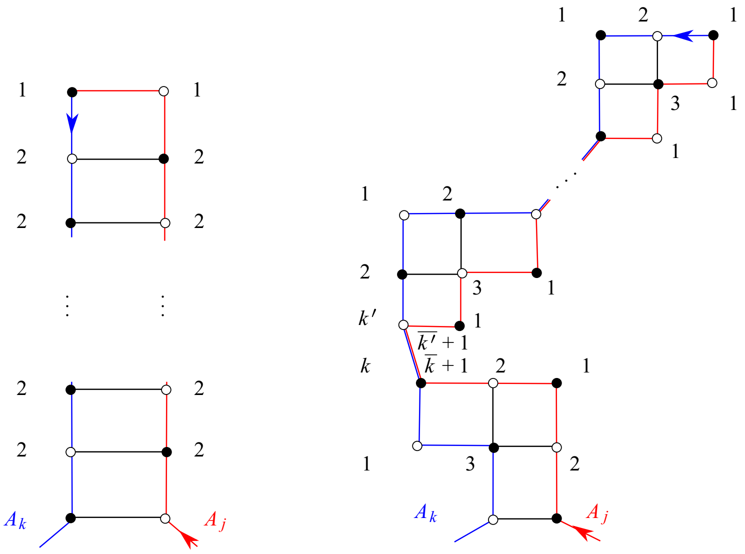

Fig. 8 Some production rules $A_i \to A_j A_k$ require an exponential number of bracket flattenings and/or spur removals to be reduced, despite A_j and A_k being already reduced

We are only considering homotopies of paths in this paper, and thus do not need the other special cases considered in [16]. Our goal is to modify the turn sequence so that it is *reduced*, i.e., contains neither spurs nor brackets. However, unlike in the aforementioned works, we cannot apply these rules directly, even inductively: in a production rule $A_i \to A_j A_k$, even when A_j and A_k are reduced, there might be an exponential number of bracket flattenings in A_i, for example in the cases in Fig. 8.

There are known techniques to handle exponentially long spurs [32, 40], but for the more intricate cases pictured in Fig. 8, especially the kind on the right side, we need to develop our own tools. In order to do that, we rely on stronger inductive forms, similarly to those used in the free homotopy test [16, 30]. A reduced path is *leftmost* (or *rightmost*, respectively) if its turn sequence contains no 1, respectively no $\overline{1}$ (we emphasize that we also use this definition in our general setting of wedges of surfaces). Note that leftmost paths become rightmost paths under reversal, and vice-versa. This trivial observation will fuel many of our arguments. One can transform a reduced path into its rightmost (resp. leftmost) form by doing *elementary right-shifts* (resp. *elementary left-shifts*) which are the three transformations $x\overline{2}^s\overline{12}^t y \to (x+1)12^{s-1}32^{t-1}1(y+1)$, $x\overline{2}^s\overline{1}y \to (x+1)12^s(y+1)$ and $x\overline{1}2^t y \to (x+1)2^t1(y+1)$ or their mirrors, see Fig. 7.

For each character in the straight-line program, we inductively compute both a rightmost and leftmost turn sequence. Then, both are used to carry the induction step. So the next step of our algorithm is the following reduction algorithm. It takes as input the straight-line program $\mathbb{A}$ describing a walk on a system of quads, and at the same time transforms it into two compressed turn sequences representing the reduced leftmost and rightmost forms for the walk. In these representations, every character

A_i of $\mathbb{A}$ encoding a walk w_i gives rise to two characters A_i^L and A_i^R representing the turn sequences of the leftmost and rightmost form of w_i. Since turn sequences only encode the turns at the interior vertices of a path and forget where the path starts, we store this information separately: for each character in the straight-line program, we store in a dictionary its starting and ending edges (which might be empty).

Reduction Algorithm

Input: A compressed walk on a quad system described by a straight-line program $\mathbb{A}$, as output by Lemma 6.2.

Output: Two compressed turn sequences on the quad system which are the leftmost and rightmost forms of the input, and their starting and ending edges.

– The leaves of the production tree are transformed to empty sequences since they consist of a single edge. We store the edge as both the starting and the ending edge in our dictionary.

– Let $A_i \rightarrow A_j A_k$ be a production rule, and assume that we have inductively computed rightmost and leftmost reduced turn sequences for A_j and A_k, which are denoted by A_j^R, A_j^L, A_k^R, and A_k^L, as well as their starting and ending edges. We explain how to compute a rightmost reduced turn sequence A_i^R, the leftmost case being symmetric. If the dictionary entries of A_j^R and A_k^R are empty, A_i^R is an empty sequence, with empty starting and ending edges. If one of the dictionary entries is empty, A_i^R is the other turn sequence, inheriting its starting and ending edges. Otherwise:

1. We look up the last edge of A_j^R and the first edge of A_k^R in the dictionary and compute the turn τ between them.[5] If it is different from 0 or 1, we go to Step 3. Otherwise, we use Lemma 2.1 to compute the largest m so that $A_j^R[-m:] = \overline{A_k^L}[-m:]$ if they have the same ending edge (let us emphasize that we use the **leftmost** form of A_k here). We use Lemma 6.3 to compute the last edge of $\overline{A_k^R}[:-m-1]$. If it arrives at the same vertex as $\overline{A_k^L}[:-m-1]$, we go to Step 2 with $A_j^R := A_j^R[:-m-1]$ and $A_k^R := A_k^R[m+1:]$. If not, we go to Step 2 with $A_j^R := A_j^R[:-m-1]$ and $A_k^R := (\tau-1)A_k^R[m+2:]$, where τ is the turn $A_k^R[m+1]$.
2. Using Lemma 6.3, we compute[5] the turn τ between the last edge of A_j^R and the first edge of A_k^R. If $\tau = 1$, we use Lemma 2.1 to compute a maximal subword $x2^{m'}112^{m'}y$ in $A_j^R \tau A_k^R$, i.e., $A_j^R \tau A_k^R = Sx2^{m'}112^{m'}yT$, and we set A_j^R to be $S(x-1)$ and A_k^R to be T and go to Step 3. It $\tau = 0$, we use Lemma 2.1 to compute the largest m so that $A_j^R[-m:] = \overline{A_k^R}[-m:]$ and go to Step 3 with $A_j^R := A_j^R[:-m-1]$ and $A_k^R := A_k^R[m+1:]$.
3. If we came directly here, we look up the last edge of A_j^R and the first edge of A_k^R in the dictionary and compute the turn τ between them.[5] If we came from

[5] At the start of Steps 1, 2, and 3, it might happen that the last letter of A_j^R and the first letter of A_k^R are both *edges*, if half of a jumping turn got removed. In that case, we remove those edges since they are superfluous.

Step 2, we compute[5] these edges using Lemma 6.3 and use them to compute τ. We concatenate the two words, adding τ in between.

a. If $\tau = \overline{1}$, we use Lemma 2.1 to compute the maximal subword $x\overline{2}^s\overline{12}^t y$, and apply an elementary shift which changes it into $(x+1)$ $12^{s-1}32^{t-1}1(y+1)$, $(x+1)12^s(y+1)$, or $(x+1)2^t1(y+1)$, depending on whether s or t is zero (if x or y are edges, they are changed into the turned edges). Since it is straightforward to encode efficiently 2^s using straight-line programs, this is done in polynomial time. We go to Step 3.b.

b. If $\tau = 1$ or $\tau = 2$, we use Lemma 2.1 to compute a maximal subword of the form $x12^k1y$. There might be multiple such subwords, we pick one of them arbitrarily. We change it into $(x-1)\overline{2}^k(y-1)$ (if x or y are edges, they are changed into the turned edges). Since it is straightforward to encode efficiently 2^k using straight-line programs, this is done in polynomial time. We loop this step while there are brackets. When there are none, we output the result and store in our dictionary the first and last edge: these are the first and last edge of respectively, A_j^R and A_k^R, perhaps turned because of one of the elementary operations, or perhaps made empty if the entire word became empty.

c. In any other case, we do not change anything and output the concatenated word, and store in our dictionary the first edge of A_j^R and the last edge of A_k^R.

Step 3 simply implements the bracket flattenings and the shifts, so the main mysteries in this algorithm are Steps 1 and 2. These are tailored to deal with the bad cases pictured in Fig. 8, which are the only possible bad cases; see the proof of Lemma 6.6. Figure 9 illustrates how Step 1 reduces (a subset of) the right case of Fig. 8 to its left case, and how Step 2 deals with that case. Note that in these pictures, A_j and A_k were already rightmost.

We claim that after this procedure, the path that represents the word A_i^R (respectively A_i^L) is homotopic to the path represented by A_i, and is rightmost (respectively leftmost). This is straightforward for the leaves of the production tree, but the rest requires some more involved analysis.

For this analysis, we employ the terminology developed by Despré and Lazarus in their analysis of the combinatorics of quad systems [11, Sect. 4], to which we refer the reader for definitions. We will rely on two important structural results [11, Thms. 8 and 10] (see also [16, 30]).

Theorem 6.4 *For a path p on a combinatorial surface of negative Euler characteristic described by a system of quads, there is a unique rightmost, respectively leftmost, path homotopic to p.*

Theorem 6.5 *Let c and d be a rightmost and a leftmost path on a combinatorial surface of negative Euler characteristic described by a system of quads. Then $c \cdot d^{-1}$ bounds a disk diagram composed of an alternating sequence of paths and quad staircases connected through their tips.*

The two theorems extend directly to our setting of wedge of surfaces and circles, since the fundamental group of a wedge of spaces is the free product of their fun-

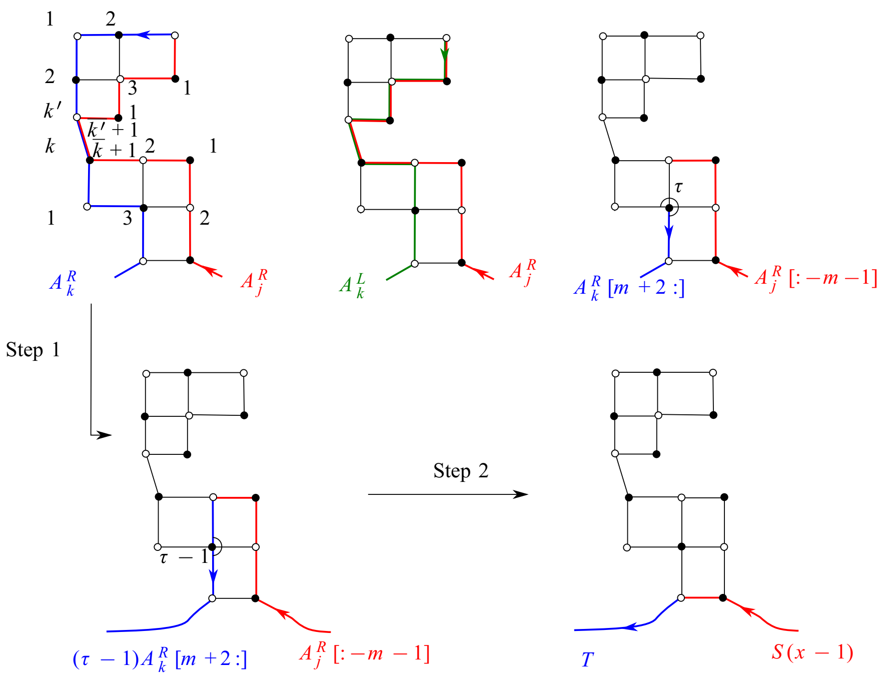

Fig. 9 The leftmost form A_k^L forms a long spur with A_j^R. After reducing A_k^R by the length of this spur, all of the staircases get removed, except possibly the last one. It disappears in Step 2

damental groups. For the first theorem, the generalization to wedges then follows by the uniqueness of the rightmost and leftmost paths in each surface they span. For the second theorem, we obtain the generalization to wedges by gluing the disk diagrams in each surface together.

The following lemma is the crux of the analysis. We state it for rightmost paths but the symmetric version with leftmost paths holds with the same proof.

Lemma 6.6 *At the end of Step 2, the paths represented by A_j^R and A_k^R are rightmost. Furthermore, denoting by τ the turn between A_j^R and A_k^R at the end of Step 2, the path represented by $A_j^R\tau A_k^R$ is homotopic to the one before these two steps, contains no spurs, contains at most two brackets, and these brackets contain both at least one* 2.

Proof Since A_j^R and A_k^R are obtained from subpaths of rightmost curves, plus possibly an additional turn, it is straightforward to verify that they are rightmost. Recall that A_i^R is the rightmost path homotopic to A_i. Since A_i^R is homotopic to $A_j^R\tau A_k^R$, the three paths represented by A_j^R, A_k^R, and $\overline{A_i^R}$ bound together a disk diagram. Since A_i^R and A_j^R are rightmost, by uniqueness of rightmost representatives of paths, they share a unique maximal subpath which is a common prefix, likewise for A_i^R and A_k^R where it is a common suffix. We consider the combinatorial triangle Δ obtained after removing these subpaths, see Fig. 10. We denote its endpoints by a, b, and c, where a is the

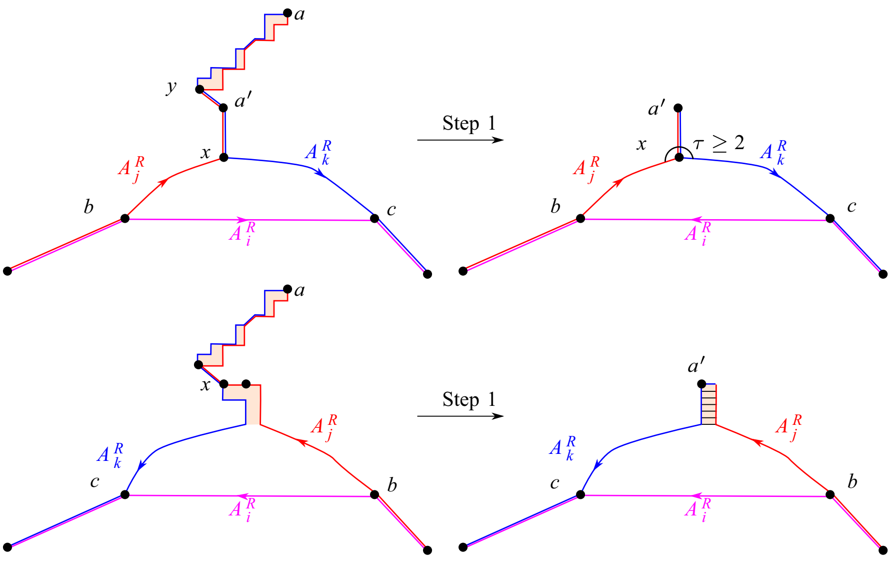

Fig. 10 The two non-degenerate cases for the combinatorial triangles in the proof of Lemma 6.6. Staircases are colored. Step 2 will remove the spur or the ladder

vertex between A_j^R and A_k^R, b the vertex between A_k^R and A_i^R, and c the last one. Let x denote the first vertex of degeneracy of Δ on the path from c to a, and y denote the vertex at the end of the first common path of A_j^R and A_k^R between x and a. Then we have a pinched disk diagram between y and a, the boundary of which is made of a subpath γ_j of A_j^R and a subpath γ_k of A_k^R. Theorem 6.5 tells us that this disk diagram is an alternating sequence of staircases and paths, which are connected by their tips. By definition, y is the tip of a staircase. We claim that y also lies on A_k^L. Otherwise, denoting by λ_1 and λ_2 the leftmost paths between a and y and between y and c, the turn between λ_1 and λ_2 at y would be $\overline{1}$, yielding a bracket with the other side of the staircase, a contradiction.

Since γ_j and γ_k are both rightmost, they have the same length and λ_1, the leftmost path homotopic to γ_k is exactly $\overline{\gamma_j}$. Therefore, during Step 1, the whole subdisk between γ_j and γ_k got removed.

We denote by Δ' the disk diagram obtained after doing Step 1, with endpoints a', b, and c, and we distinguish cases depending on its orientation. If Δ' is degenerate, i.e., has no quads in its interior, then after removing possibly one sequence of spurs in Step 2, we have $A_j^R \tau A_k^R = A_i^R$ and thus we have successfully computed a rightmost form for A_i.

If the boundary of Δ', oriented $c \to a' \to b \to c$, is oriented counter-clockwise, then a' is on the path between x and y, since A_k^L and A_j^R can not coincide below x, see Fig. 10, top. After removing the sequence of spurs between a' and x in Step 2, the (positive) turn between A_j^R and A_k^R is at least 2, since otherwise there would be a spur in A_j^R or A_k^R.

If the boundary of Δ', oriented $c \to a' \to b \to c$, is oriented clockwise, we observe that x also lies on A_k^L. Indeed, otherwise, since A_k^L is at distance at most one from A_k^R and Δ' is not degenerate, there would be a common vertex z that would be both on A_j^R and A_k^L and such that the paths from z to a are different, since one contains x and not the other. This would contradict uniqueness of canonical paths. Therefore, the entire pinched disk between x and a has been removed in Step 1. Now we look at the maximal (possibly empty) staircase having a' as a tip and being bounded by pair of subpaths α_j and α_k of A_j^R and A_k^R, except for one edge. If there is no such staircase, nothing happens in Step 2, and we are done. If there is such a staircase, we claim that most of it got already removed when we get to Step 2. If α_k contains one or more 1s, then the start of the leftmost form of α_k coincides with the start of α_j^{-1}, and thus this subword containing this 1 has also been eliminated during Step 1, see Fig. 9. Note that the slight offset of the indices in Step 1 is due to the difference in length between α_j and α_k, and that the new turn τ that we introduce is the missing edge to close the staircase. Therefore, α_k contains no 1, and the last turn of α_j is a 1. Therefore the entire staircase is a ladder, as pictured in the left of Fig. 8 and on the bottom right of Fig. 10, which gets removed during Step 2.

Let τ denote the turn between A_j^R and A_k^R after Steps 1 and 2. Then in the clockwise case, τ is at least 2 and thus at most one bracket can exist in $A_j^R \tau A_k^R$. In the counter-clockwise case, the only brackets that can exist in $A_j^R \tau A_k^R$ must contain a 2, as otherwise the staircase of the previous paragraph would not have been maximal. There are at most two such brackets: one finishing at τ and one starting at τ, since otherwise such a bracket would have been present in A_j^R or A_k^R. As the reductions correspond to flattening brackets and removing spurs on a disk diagram, they are homotopies. Finally, since the degenerate parts of the disk diagram have been removed, there are no spurs. □

We now prove that the reduction algorithm does indeed compute rightmost and leftmost forms.

Lemma 6.7 *The* Reduction Algorithm *runs in polynomial time. The straight-line program that it outputs has the property that every pair of characters A_i^R and A_i^L encode a pair of turn sequence corresponding respectively to the rightmost and leftmost paths homotopic to A_i. In particular, at the top level, $\mathbb{A}^R$ and $\mathbb{A}^L$ are the rightmost and leftmost paths homotopic to the compressed input walk.*

Proof By Lemma 6.6, when the algorithm reaches Step 3, the path $A_j^R A_k^R$ may not be reduced for two reasons: either the turn between A_j^R and A_k^R is a $\overline{1}$, or there is one or two brackets with at least one 2 between them. Note that the two cases are exclusive. We argue that no spurs or brackets remain after the two reductions 3.a and 3.b. For the first one (reduction a), note that $x, y \neq \overline{2}$ by definition and x and y are neither 0 nor $\overline{1}$ because A_j^R is rightmost. We may have $x = 1$ or $y = 1$, which may create at most two brackets, but note that these brackets are disjoint and contain each at least one 2.

When we reach Step 3.b, we therefore have at most two brackets, each containing at least one 2. We conclude by proving that no new brackets nor spurs are created during an application of this Step 3.b. Note that x and y cannot be $\overline{1}$ because the curves are rightmost, cannot be 0 because they contain no spurs, and cannot be 1 because otherwise there would be a bracket without a 2. They could be 3, but the $\overline{2}$ blocks the resulting 2 from participating in a bracket since $\overline{2} \neq 2$ as there is no torus surface among the wedge summands. Therefore, the only way for $x-1$ or $y-1$ to be involved in a bracket is when one of them is 1, in which case it bounds a second bracket that already existed before the reduction. After reducing that one, there are no more possible spurs nor brackets. In particular, Step 3.b only loops a constant number of times, and thus the whole inductive step is polynomial. In turn, computing the full straight-line programs for both reduced words $\mathbb{A}^L$ and $\mathbb{A}^R$ is also polynomial. □

We are now ready to prove Theorem 6.1.

Proof of Theorem 6.1 Using Lemma 6.2, we first turn our walk w into a walk on a quad system. Then, using Lemma 6.7, we compute a compressed rightmost reduced loop w' homotopic to w in this quad system. By Theorem 6.4, there is a unique such rightmost loop. Combined with the data of the first and last edge, this is our canonical form. If w is homotopic to a trivial walk, it is homotopic to the empty loop, which is rightmost. Therefore, to test contractibility, it suffices to test whether w' is the empty word. □

Finally, let us comment on how to adapt this algorithm in the case where there are tori in the wedge. In tori summands, we dispense from using turn sequences and simply reduce to a one-vertex one-face graph, which therefore has two edges a and b. Homotopy in tori is a simple matter since the fundamental group is $\mathbb{Z}^2$, corresponding exactly to how many times the oriented edges a and b are taken. Therefore, we directly use for our encoding a pair (k,ℓ) representing this, with the convention that $\overline{(k,\ell)} = (-k,-\ell)$. The reduction algorithm has an additional rule stipulating that $(k_1,\ell_1)(k_2,\ell_2)$ should be reduced to $(k_1+k_2,\ell_1+\ell_2)$. Any reduction to $(0,0)$ triggers a search for a deeper sequences of spurs and staircases as in Steps 1 to 3.

Proofs of the Main Theorems

We now have all the tools to prove our main theorems. Theorem 1.3 is a subcase of Theorem 6.1 when the the wedge of surfaces consists of a single surface.

Proof of Theorem 1.2 We can make Δ reduced in polynomial time [15]. By Proposition 4.1, one can compute in polynomial time a new triangulation of the surface S so that the multi-curve Δ has polynomial complexity and the curve γ is encoded as a straight-line program. By Proposition 5.1, we can compute a wedge of surfaces and circles K and a walk w on K so that w is contractible in K if and only if γ belongs to the normal subgroup generated by the curves Δ in S. The simplicial complex K can be transformed to a wedge of combinatorial surfaces and circles easily. Then the theorem follows from Theorem 6.1. □

Proof of Theorem 1.1 By Proposition 3.2, we can reduce our problem to the case of orientable manifolds. Note that the size of the manifold at most doubles in this process.

Then, Proposition 3.1 reduces the problem to the disjoint normal subgroup membership problem. The **NP** certificate consists of the certificate in that proposition, and the corresponding algorithm is to guess this certificate. The disjoint normal subgroup membership problem is then solved using Theorem 1.2. □

Acknowledgements The authors would like to thank Mark Bell, Ben Burton, and Jeff Erickson for helpful discussions. We also thank an anonymous referee of [9] (which is a heavily modified version of [8]) for suggesting the use of a maximal compression body as a simple exponential-time algorithm for deciding contractibility of arbitrary (non-compressed) curves on the boundary of a 3-manifold. We are also grateful to the anonymous reviewers for their careful reading of the manuscript which greatly improved the paper.

References

1. Agol, I., Hass, J., Thurston, W.: The computational complexity of knot genus and spanning area. Trans. Am. Math. Soc. **358**(9), 3821–3850 (2006)
2. Aschenbrenner, M., Friedl, S., Wilton, H.: Decision problems for 3-manifolds and their fundamental groups. In: Interactions Between Low-Dimensional Topology and Mapping Class Groups (Bonn 2013). Geometry & Topology Monographs, vol. 19, pp. 201–236. Geom. Topol. Publ., Coventry (2015)
3. Aschenbrenner, M., Friedl, S., Wilton, H.: 3-Manifold Groups. EMS Series of Lectures in Mathematics. European Mathematical Society, Zürich (2015)
4. Bachman, D., Derby-Talbot, R., Sedgwick, E.: Computing Heegaard genus is NP-hard. In: A Journey Through Discrete Mathematics, pp. 59–87. Springer, Cham (2017)
5. Bell, M.C.: Simplifying triangulations. Discrete Comput. Geom. **66**(1), 1–11 (2021)
6. Burton, B.A.: A new approach to crushing 3-manifold triangulations. Discrete Comput. Geom. **52**(1), 116–139 (2014)
7. Burton, B.A., Colin de Verdière, É., de Mesmay, A.: On the complexity of immersed normal surfaces. Geom. Topol. **20**(2), 1061–1083 (2016)
8. Colin de Verdière, É., Parsa, S.: Deciding contractibility of a non-simple curve on the boundary of a 3-manifold. In: 28th Annual ACM-SIAM Symposium on Discrete Algorithms (Barcelona 2017), pp. 2691–2704. SIAM, Philadelphia (2017)
9. Colin de Verdière, É., Parsa, S.: Deciding contractibility of a non-simple curve on the boundary of a 3-manifold: a computational Loop Theorem (2020). arXiv:2001.04747
10. Dehn, M.: Transformation der Kurven auf zweiseitigen Flächen. Math. Ann. **72**(3), 413–421 (1912)
11. Despré, V., Lazarus, F.: Computing the geometric intersection number of curves. J. ACM **66**(6), #45 (2019)
12. Dynnikov, I.: Counting intersections of normal curves. J. Algebra **607**(B), 181–231 (2021)
13. Dynnikov, I., Wiest, B.: On the complexity of braids. J. Eur. Math. Soc. **9**(4), 801–840 (2007)
14. Epstein, D.B.A., Cannon, J.W., Holt, D.F., Levy, S.V.F., Paterson, M.S., Thurston, W.P.: Word Processing in Groups. Jones and Bartlett, Boston (1992)
15. Erickson, J., Nayyeri, A.: Tracing compressed curves in triangulated surfaces. Discrete Comput. Geom. **49**(4), 823–863 (2013)
16. Erickson, J., Whittlesey, K.: Transforming curves on surfaces redux. In: 24th Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans 2013), pp. 1646–1655. SIAM, Philadelphia (2013)
17. Hagenah, Ch.: Gleichungen mit regulären Randbedingungen über freien Gruppen. PhD thesis, Universtät Stuttgart (2000). https://elib.uni-stuttgart.de/handle/11682/2469
18. de la Harpe, P.: Topics in Geometric Group Theory. Chicago Lectures in Mathematics, University of Chicago Press, Chicago (2000)
19. Hass, J., Lagarias, J.C., Pippenger, N.: The computational complexity of knot and link problems. J. ACM **46**(2), 185–211 (1999)
20. Hass, J., Snoeyink, J., Thurston, W.P.: The size of spanning disks for polygonal curves. Discrete Comput. Geom. **29**(1), 1–17 (2003)
21. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
22. Hempel, J.: 3-Manifolds. AMS Chelsea Publishing, vol. 349. American Mathematical Society, Providence (2004)

23. Holt, D., Lohrey, M., Schleimer, S.: Compressed decision problems in hyperbolic groups. In: 36th International Symposium on Theoretical Aspects of Computer Science (Berlin 2019). Leibniz International Proceedings in Informatics, vol. 126, # 37. Leibniz-Zent. Inform., Wadern (2019)
24. Jaco, W., Rubinstein, J.H.: 0-efficient triangulations of 3-manifolds. J. Differ. Geom. **65**(1), 61–168 (2003)
25. Jaco, W., Tollefson, J.L.: Algorithms for the complete decomposition of a closed 3-manifold. Ill. J. Math. **39**(3), 358–406 (1995)
26. Kuperberg, G.: Algorithmic homeomorphism of 3-manifolds as a corollary of geometrization. Pacific J. Math. **301**(1), 189–241 (2019)
27. Lackenby, M.: Some conditionally hard problems on links and 3-manifolds. Discrete Comput. Geom. **58**(3), 580–595 (2017)
28. Lackenby, M.: Algorithms in 3-manifold theory (2020). arXiv:2002.02179
29. Lackenby, M.: The efficient certification of knottedness and Thurston norm. Adv. Math. **387**, #107796 (2021)
30. Lazarus, F., Rivaud, J.: On the homotopy test on surfaces. In: 53rd Annual IEEE Symposium on Foundations of Computer Science (New Brunswick 2012), pp. 440–449. IEEE Computer Society, Los Alamitos (2012)
31. Löffler, M., Lubiw, A., Schleimer, S., Wolf Chambers, E.M. (eds.): Computation in low-dimensional geometry and topology (Dagstuhl Seminar 19352). Dagstuhl Reports **9**(8), 84–112 (2019)
32. Lohrey, M.: Word problems and membership problems on compressed words. SIAM J. Comput. **35**(5), 1210–1240 (2006)
33. Lohrey, M.: The Compressed Word Problem for Groups. SpringerBriefs in Mathematics. Springer, New York (2014)
34. Matveev, S.: Algorithmic Topology and Classification of 3-Manifolds. Algorithms and Computation in Mathematics. vol. 9. Springer, Berlin (2007)
35. de Mesmay, A., Rieck, Y., Sedgwick, E., Tancer, M.: The unbearable hardness of unknotting. In: 35th International Symposium on Computational Geometry (Portland 2019). Leibniz International Proceedings in Informatics, vol. 129, # 49. Leibniz-Zent. Inform., Wadern (2019)
36. Mohar, B., Thomassen, C.: Graphs on Surfaces. Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore (2001)
37. Moise, E.E.: Affine structures in 3-manifolds. V. The triangulation theorem and Hauptvermutung. Ann. Math. **56**, 96–114 (1952)
38. Schaefer, M., Sedgwick, E., Štefankovič, D.: Algorithms for normal curves and surfaces. In: Computing and Combinatorics (Singapore 2002). Lecture Notes in Computer Science, vol. 2387, pp. 370–380. Springer, Berlin (2002)
39. Schaefer, M., Sedgwick, E., Štefankovič, D.: Computing Dehn twists and geometric intersection numbers in polynomial time. In: 20th Canadian Conference on Computational Geometry (Montreal 2008), pp. 111–114 (2008)
40. Schleimer, S.: Polynomial-time word problems. Comment. Math. Helv. **83**(4), 741–765 (2008)
41. Schubert, H.: Bestimmung der Primfaktorzerlegung von Verkettungen. Math. Z. **76**, 116–148 (1961)
42. Stillwell, J.: Classical Topology and Combinatorial Group Theory. Graduate Texts in Mathematics, vol. 72. Springer, New York (1993)
43. Waldhausen, F.: The word problem in fundamental groups of sufficiently large irreducible 3-manifolds. Ann. Math. **88**, 272–280 (1968)

Authors and Affiliations

Erin Wolf Chambers[1] · **Francis Lazarus**[2] · **Arnaud de Mesmay**[3] · **Salman Parsa**[1]

Erin Wolf Chambers
erin.chambers@slu.edu

Francis Lazarus
francis.lazarus@grenoble-inp.fr

Arnaud de Mesmay
arnaud.de-mesmay@univ-eiffel.fr

Salman Parsa
salman.parsa@slu.edu

1 Saint Louis University, Saint Louis, MO, USA

2 G-SCOP, CNRS, UGA, Grenoble, France

3 LIGM, CNRS, Univ. Gustave Eiffel, ESIEE Paris, 77454 Marne-la-Vallée, France