Journal of Knot Theory and Its Ramifications, Vol. 11, No. 4 (2002) 601–645

TOWARDS AN IMPLEMENTATION OF THE B–H ALGORITHM FOR RECOGNIZING THE UNKNOT

JOAN S. BIRMAN
Dept. of Mathematics, Barnard College
Mail Code 4427, Columbia University, 2990 Broadway, New York, N.Y., USA 10027

MARTA RAMPICHINI
Dipartimento di Matematica, Università di Milano
via Saldini 50, 20133 Milano MI, Italy

PAOLO BOLDI and SEBASTIANO VIGNA
Dipartimento di Scienze dell'Informazione, Università di Milano
via Comelico 39/41, 20135 Milano MI, Italy

ABSTRACT

In the manuscript [2] the first author and Michael Hirsch presented a then-new algorithm for recognizing the unknot. The first part of the algorithm required the systematic enumeration of all discs which support a 'braid foliation' and are embeddable in 3-space. The boundaries of these 'foliated embeddable discs' (FED's) are the collection of all closed braid representatives of the unknot, up to conjugacy, and the second part of the algorithm produces a word in the generators of the braid group which represents the boundary of the previously listed FED's. The third part tests whether a given closed braid is conjugate to the boundary of a FED on the list. In this paper we describe implementations of the first and second parts of the algorithm. We also give some of the data which we obtained. The data suggests that FED's have unexplored and interesting structure. Open questions are interspersed throughout the manuscript. The third part of the algorithm was studied in [3] and [4], and implemented by S.J. Lee [20]. At this writing his algorithm is polynomial for $n \leq 4$ and exponential for $n \geq 5$.

1. Introduction

1.1. *Background*

The subject of this paper is the question: given a knot K, can we decide whether K is the unknot? The problem was solved affirmatively by W. Haken in a groundbreaking paper published in 1961 [10]. However, showing that an algorithm exists does not mean that there is necessarily an algorithm which will be useful in practice, even for the simplest examples. Thus in 1993 (over 30 years after Haken did his work) when Hoste, Thistlethwaite and Weeks tabulated the 1,701,936 prime knots

with ≤ 16 crossings [14] they had all the tools of the trade available to them, but used a ragbag of diagrammatic techniques to eliminate unwanted appearances of the unknot. In that regard it should be noted that knot diagrams with at most 16 crossings do not even begin to exhibit the pathology which one knows exists in the general case. For example, see [8] for some examples which show why the diagrammatic approach was abandoned in the 1930's. (On the other hand, see [12] for a recent proof that an upper bound exists for the number of Reidemeister moves which must be tested to be sure that a knot diagram with a given number of crossings is not the unknot.)

Haken's work begins by constructing a triangulation of the complement of K. He then applies the theory of normal surfaces, due to Kneser [18], who showed that any surface F of minimal genus with boundary K can be assumed to be in a special position in which it intersects each tetrahedron T_i in the triangulation in an especially nice way, namely as a set of parallel sheets, each sheet being a polygonal disc whose boundary has 3 or 4 edges in the faces of ∂T_i. The polygonal discs are used to set up a system of linear equations. Solving the system allows one to decide whether, in fact, the solution set includes a normal surface which is a disc.

A very different approach to the unknot recognition problem was discovered by the first author and M. Hirsch, who developed in [2] the algorithm which is the subject of investigation in this paper. The basic idea behind the B–H algorithm rests in the braid foliation techniques of Birman and Menasco (see [6] and [1]). Braid foliations allow one to generate, in a systematic manner, a list of all of the foliated embedded discs whose boundaries are closed braid representatives of the unknot. The list is ordered by a complexity function which depends on properties of the foliated discs. One then compares a given example K with the examples on the list in order to decide whether K is the unknot.

In this paper we will give a computer implementation of certain parts of the algorithm in [2], namely the problems of enumerating the foliated embeddable discs and finding the braid words which describe their boundaries. We note that for braid index 2 the problem is trivial. For braid index 3 the unknot recognition problem was solved in [21], where it was proved that there are precisely 3 conjugacy classes of closed 3-braid representatives of the unknot. For $n = 4$ the question is much harder, because of the example in [24] and the others which are presented here. See [7] for a proof that there are infinitely many distinct conjugacy classes of 4-braid representatives of the unknot. We were able to obtain non-trivial data for braid index $n = 4$. The examples which we found have braid word descriptions with ≤ 11 band generators, however there may well be shorter braid words for the same examples. We also give a small amount of scattered data for higher braid index. We note that for $n = 4$ polynomial-time algorithms exist for the solution to the conjugacy problem and the shortest word problem [17], which could easily be integrated with our work, however we did not make a systematic attempt to do that.

The data which we obtained is given in Section 7 of this paper.

We conjecture that a practical polynomial time algorithm exists which will solve the unknot recognition problem in the special case of knots of braid index 4. Theorem 4.3 of the review article [1] (which gives a new proof of the main result in [6]) would surely play an important role in any such solution, as would the polynomial-time algorithms of [17]. The chief obstacle, as we see it, is to find an efficient way to enumerate all the foliated embeddable discs with N negative vertices and $4+N$ positive vertices. We believe that when the structure of these foliated embeddable discs is better understood, this problem will be solved.

1.2. *A review of braid foliations*

In this section we briefly review the main results of [2]. A good reference for a survey on braid foliations is [1]. After completing our review of the results which we need from [2] and [1] we explain in a precise way what we do in this paper.

The underlying plan is the following: the unknot is the unique knot which bounds a disc embedded in $\mathbf{R}^3$. All discs embedded in $\mathbf{R}^3$ can be isotoped in such a way that the boundary is a closed braid relative to the z-axis. All these embeddings can be described by a finite set of combinatorial data, and they can be listed in order of increasing complexity. To each disc we will show how to associate the braid whose closure is the boundary of the disc. So if we want to know whether a given knot K is the unknot, we first represent it as a closed braid $\hat{\beta}$, using our preferred algorithm (see [23, 28, 30]). Then compare this braid with our list of braids which are the boundary of an embedded disc, looking for a braid γ which is in the same conjugacy class as the given braid β. To check conjugacy, use for instance the algorithm in [3] and [4], as implemented by S.J. Lee [20]. In [2] an upper bound is given for the complexity of the disc to look for in the list, so the process is finite. The problem of improving that upper bound will not be considered here.

To implement our algorithms we have used [GAP 99] The GAP Group, GAP — Groups, Algorithms, and Programming, Version 4.1; Aachen, St Andrews, 1999.

Let us fix the z-axis A of $\mathbf{R}^3$ as braid axis, and the standard fibration by half-planes of the complement of A by half-planes

$$H_\theta = \{(\rho\cos\theta, \rho\sin\theta, z) \mid \rho > 0, z \in \mathbf{R}\}$$

Theorem 1 (cf Theorem 2.1 of [2]). A disc D embedded in the standard fibration of $\mathbf{R}^3 \setminus \mathbf{A}$, with boundary a closed braid $\hat{\beta}$, $\beta \in B_n$, can always be put in general position so that the induced singular foliation on D has the following properties:

1. All intersections of the disc D with the axis A are transversal. These intersections consist of P positive and N negative points, where the sign is positive if the orientation of A agrees with that of D, otherwise negative. We call these $P+N$ intersection points 'vertices'. The braid index is $n = P - N$.

2. The disc D intersects almost all half-planes H_θ transversally, in what we call regular leaves.

3. The foliation in a neighborhood of ∂D is transverse.

4. The foliation in a small circular neighborhood of each vertex is radial.

5. There are a finite number $P + N - 1$ of singular half-planes H_{θ_i} to which D is tangent in one point, which is a non-degenerate saddle. A saddle together with its four leaves (branches) is called a singular leaf.

6. The regular leaves can be of the following two types: type a: a simple arc with one endpoint on ∂D and the other on A; type b: a simple arc with both endpoints on A.

7. The saddles are restricted to the following types:

   aa-saddles: singularities between two a-arcs;

   ab-saddles: singularities between an a-arc and a b-arc;

   bb-saddles: singularities between two b-arcs.

   Each saddle can be either positive or negative, according as the orientation of the disc and the tangent half-plane at the saddle point agree or disagree.

8. The vertices are cyclically ordered along A, and the saddles are cyclically ordered around A.

The code for a foliated disc: It is not explicitly explained in [2] how to represent an embeddable disc on a computer: we will encode the description of an embedded disc D in terms of its vertices and saddles as follows:

The vertex string V is the list of (P, N) vertices; each positive vertex will be denoted by an integer number $k \in \{1, 2, \ldots P\}$; each negative vertex will be denoted by a pair of integer numbers $k.j$, where k is the number of the immediately preceding positive vertex in V (k might be 0) and j is the ordinal number of the negative vertex in the subset of negative vertices between k and $k + 1$;

The ordered list of saddles, in which each saddle will be denoted by: the list of vertices involved in the saddle: two positive for an aa-saddle, two positive and one negative for an ab-saddle, two positive and two negative for a bb-saddle; and the sign, ± 1.

For a fixed braid index n we can assign to a disc D its **complexity** (P, N), that is the number of positive and negative vertices, with $P - N = n$, and list all embeddable discs in order of increasing complexity. This is possible thanks to the following theorems:

Theorem 2 (cf Theorem 2.2 of [2]). The combinatorial data for an embeddable disc D, i.e. the cyclically ordered list of vertices, with their signs, and the cyclically ordered list of saddles with their signs, determine the embedding in $\mathbf{R}^3$, uniquely up to foliation-preserving isotopy. They also determine the embedding of the boundary of the disc as a closed braid.

To see an example of the singular foliation on an embeddable disc, look ahead to Figure 6: We show there a disc D in which we have drawn all vertices and saddles, with all singular leaves. The complement of the singular leaves in the disc is the disjoint union of open discs, some of them bounded by: an arc of ∂D, a positive vertex and some singular leaves (there are 18 like this in Figure 6); the others (maybe zero) bounded by one positive and one negative vertex, and some singular leaves (there are 13 like this in Figure 6). Discs of the first type can be foliated by regular leaves which are a-arcs, with endpoints on the positive vertex and ∂D. Discs of the second type can be foliated by regular leaves which are b-arcs, with endpoints on the positive and the negative vertex.

The definition of an extended boundary word can be found in Section 2.

Theorem 3 (cf Theorem 3.4 of [2]). From the set of combinatorial data describing an embeddable disc D we extract a unique extended boundary word, which represents a braid whose closure is the link consisting of the boundary of D and the N unlinked small circles bounding small disc neighborhoods of the N negative vertices of D.

In section 5 we will give an explicit implemented algorithm to get the boundary braid from the extended boundary braid.

For a given set of combinatorial data as described before we have to test embeddability. For this purpose, let us give some necessary definitions:

Definition 1 (cf [2]). A **b-arc** is a regular leaf $b(i, j.h)$ which is a simple arc connecting a positive vertex i to a negative vertex $j.h$. A **generalized or gb-arc** is either a b-arc or the part of a singular leaf of an aa-saddle connecting the two positive vertices and passing through the saddle point.

Since the foliation around each vertex is the standard radial foliation, we can distinguish leaves around each vertex by means of their angle θ. If there exists at some θ a b-arc $b(i, j.h)$, then there exists a maximal interval (θ_l, θ_k) in which all regular leaves around i and $j.h$ are $b(i, j.h)$. In this case we say that the b-arc exists in (θ_l, θ_k). Say that the gb-arc $gb(i, j)$ exists in (θ_{k-1}, θ_k) if there is an aa-saddle (i, j) occuring at θ_k. For instance, looking at Figure 9 see $b(3, 0.1)$ in (θ_{14}, θ_7) (leaves go clockwise around a negative vertex) and $gb(4, 8)$ in (θ_5, θ_6).

Theorem 4 (Theorem 3.5 of [2]). A disc D, given in terms of combinatorial data, is embeddable if and only if it satisfies the following three conditions:

1. The saddles about each positive (respectively negative) vertex are in counterclockwise (respectively clockwise) order;

2. The vertices attached to a positive (respectively negative) saddle are in counterclockwise (respectively clockwise) order;

3. The endpoints of a $(g)b$-arc in (θ_{k-1}, θ_k) never separate the endpoints of a b-arc in the same interval.

In what follows the expression 'a disc with (P, N) vertices' will mean a disc (or its combinatorial description) with P positive and N negative vertices.

The fundamental tool for listing all embeddable discs in [2] is the **insertion of an ab-tile**: given an embeddable disc D with (P, N) vertices, Birman and Hirsch describe how to get from D a new disc D' with $(P, N+1)$ vertices and one more saddle of type ab.

Theorem 5 (cf Theorem 4.1 of [2]). Each embeddable disc D with (P, N) vertices can be constructed by starting from an embeddable disc D_0 with $(P, 0)$ vertices, adding N ab-tiles one at a time. At each stage the new negative vertex and the new ab-saddle are inserted into the order of the older vertices and saddles, in such a way that the new disc is embeddable.

Theorem 6 (Theorem 4.2 of [2]). All possible embeddable discs of fixed braid index n may be enumerated in order of increasing (P, N), with $P - N = n$, by the following (not necessarily efficient) procedure:

- enumerate all possible discs with $(n, 0)$ vertices, testing each for embeddability; discard non embeddable ones;

- enumerate all discs with $(n + j, 0)$ vertices, testing each for embeddability; discard non embeddable ones; then add j ab-tiles in all possible ways, testing each obtained disc for embeddability; discard non embeddable ones; get all embeddable discs with $(n + j, j)$ vertices.

We will call discs with zero negative vertices **positive discs**. Since we are interested in conjugacy classes of braids, we have to notice that this list has duplicates, because non isotopic embeddable discs may have the same boundary braid, or different boundary braids in the same conjugacy class.

In what follows we will show:

- how our set of combinatorial data describes an (embeddable) disc;

- how to translate many properties of the combinatorial foliation in braid words in the band generators;

- how to enumerate all positive embeddable discs;

- how to reduce a lot of the redundancy in the resulting list;

- how to insert ab-tiles and test embeddability with a computer program;

- how to find the boundary word of an embeddable disc;

- how to fill the list via a different way: no more by insertion of ab-tiles and embeddability test, but via enumeration of sequences of half-planes completely describing embeddable discs.

The main reason for wanting to see the data produced by an algorithm is if it suggests new structure which will then lead to a better algorithm. For this reason we will intersperse "questions" throughout the manuscript, as they occur to us in the context of our work. Some of them have immediate answers, but most relate to open problems.

2. The disc, its code and the associated braid word

We will use the band presentation of the braid group which is given in [3]. The generators are the $\binom{n}{2}$ braids $\{(i,j), n \geq i > j \geq 1\}$, such that the i^{th} strand crosses over the j^{th} strand, with all the other strands left unchanged under these two. The relations are of two types:

$$(i,j)(k,l) = (k,l)(i,j), \text{ when } (i-k)(i-l)(j-k)(j-l) > 0,$$

(the condition means that the two pairs of indices are non interlocking), and

$$(i,j)(j,k) = (i,k)(i,j) = (j,k)(i,k), \text{ when } n \geq i > j > k \geq 1.$$

We will denote the inverse of a generator (i,j) by $\overline{(i,j)}$.

Suppose we are given a list of combinatorial data for an embeddable disc. In this section we explain how to draw the singular foliation of the disc, and how to associate to D its extended boundary word. Let us consider the following example:

$$D = \{[[4.1,6,7],1],[[9,10],1],[[8.2,10,11],1],[[4,8.1,11],1],[[4,8],1],$$
$$[[0.1,3,7],-1],[[0.1,7,8],-1],[[0.2,2,4.1,6],-1],[[0.2,6,7],-1],[[5,6],1],$$
$$[[1,5],1],[[0.2,2,4.1,7],1],[[0.1,3,8],1],[[4,8.1,11],-1],[[8.2,10,11],-1]\}.$$

Definition 2 (cf [2]). For an embeddable disc D with (P,N) vertices, its **extended boundary word** is the braid word in the band generators of B_P with each letter given by the pair of positive vertices and the sign of the corresponding saddle of D.

In our example,

$$EW(D) = (7,6)(10,9)(11,10)(11,4)(8,4)\overline{(7,3)(8,7)(6,2)}\cdot$$
$$\cdot\overline{(7,6)}(6,5)(5,1)(7,2)(8,3)\overline{(11,4)(11,10)}.$$

Proposition 1. If a word W of a braid in B_P is the extended boundary word of an embeddable disc with (P,N) vertices, then it has the following properties:

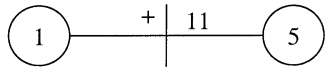

Fig. 1. The first saddle.

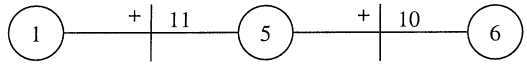

Fig. 2. The first two saddles.

1. The length of W (in band generators) is $P + N - 1$;
2. The induced permutation $\rho(W) \in S_P$ is a product of one $(P - N)$-cycle and N 1-cycles.

Proof. To each saddle of the disc corresponds one letter of W, in the same order, to each positive vertex corresponds one index, and to each negative vertex corresponds one single unknotted strand of the closed braid, unlinked from the rest. The first condition comes from the Euler characteristic of the foliated disc. □

In our example $EW \in B_{11}$, so that $P = 11$; its length is 15, therefore $N = 5$ and $P - N = 6$; the associated permutation is

$$\rho(EW) = (1, 5, 6, 8, 10, 9)(2)(3)(4)(7)(11).$$

Now we can explain how to draw leaves of the foliation on the disc. Consider the first index of the $(P - N)$-cycle, say j_1: look for the first letter of $EW(D)$ in which that index occurs: say $(j_1, j_2)^{\pm 1}$ (or $(j_2, j_1)^{\pm 1}$). Draw the two positive vertices and the singular leaves joining them, and label the vertices with their numbers and the saddle with its sign and number (see Figure 1). Then look for the (cyclically) next letter containing j_2: if it is a different one, then draw it attached to the previous one (see Figure 2), if it is the same letter, then procede to the next letter containing j_1. We have to run twice through each aa-saddle.

To respect embeddability, when a positive vertex has two or more saddles attached, and we have to draw another one attached to it, we must put it in the right (counterclockwise) cyclic order about the vertex (see Figure 3).

Procede as indicated above until you get back to j_1 and the $(P - N)$-cycle is completed. These singular leaves divide the disk into an outer part, which is connected to the boundary, and some inner parts, in which the negative vertices lie along with possibly those positive vertices which at this stage have not yet been drawn (see Figure 4). They are vertices occurring in the 1-cycles of the permutation. For each of them, draw its cycle of saddles in a similar way (see Figure 5), then attach it inside the appropriate inner region.

It remains to add the negative vertices. Look at the code of the disc, and put each negative vertex in its place, joining it to the (already existing) saddles by the other singular leaves (see Figure 6).

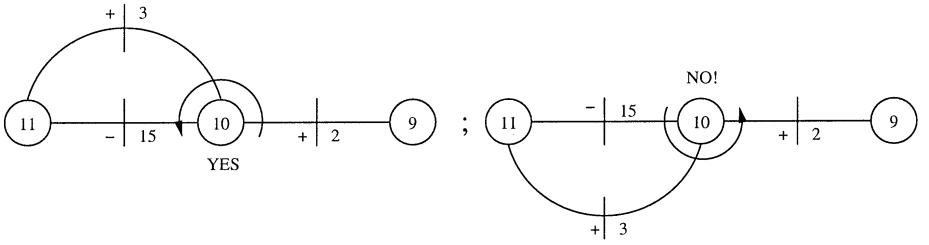

Fig. 3. Respect order about each vertex.

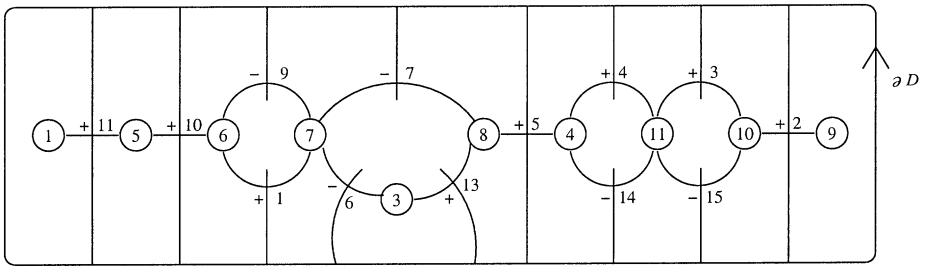

Fig. 4. The disc with all its 'external' saddles.

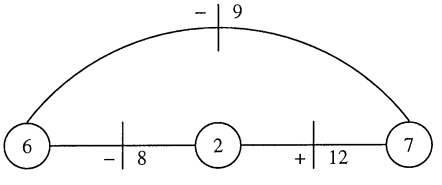

Fig. 5. The cycle of saddles about another positive vertex.

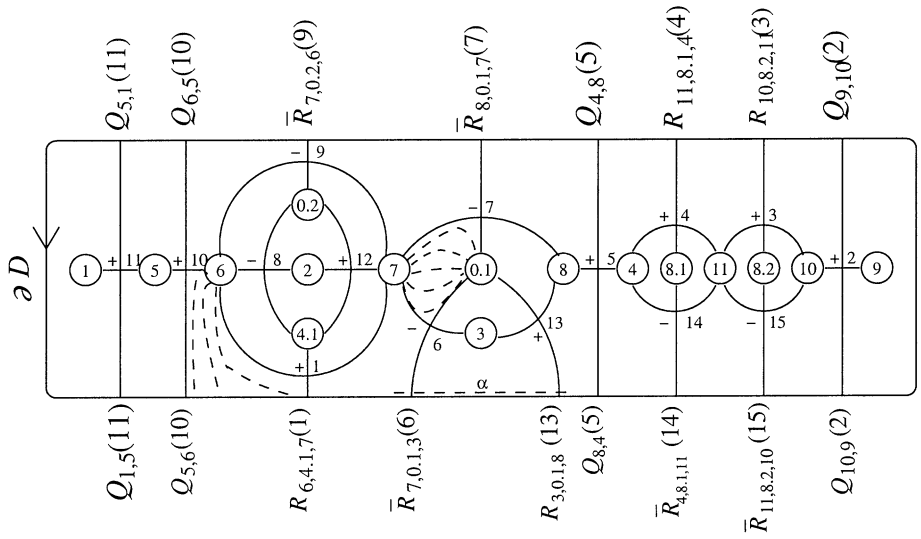

Fig. 6. The foliated disc and the code for its boundary.

Remark: Notice that this process of drawing a disc can be performed almost entirely from the extended boundary word. Only at the end do we need to know the exact position of the negative vertices in the vertex string. But their number N and their topological position inside the inner regions of the foliation are already specified by EW. In some cases it is also possible to decide their position in the vertex string simply by reading their relative order about the saddles to which they are attached.

Question 1. It's natural to ask whether every word which satisfies the conditions of Proposition 1 actually can be realized by an embeddable foliated disc? The answer is 'no'. For example $W = (3,2)(4,1)(3,1)(4,3)$ satisfies the conditions of Proposition 1 but the corresponding disc is not embeddable. To see this consult Figure 7. For saddle 2 we must have $4 < v < 1$, but for saddle 3 we require $1 < v < 3$, which is impossible.

If $EW = (6,5)(5,4)\overline{(4,2)}(3,1)(5,3)(4,2)$, we can draw all the disc and see that the negative vertex must be either 0.1 or 5.1. But neither of them is embeddable: they both pass the first part of embeddability test, but not the b-arc test: in the first case, in $(4,5)$ we have $gb(3,5)$ interlocking with $b(4,0.1)$; in the second case we have for instance in $(3,4)$ $gb(3,5)$ interlocking with $b(4,5.1)$.

Question 2. Are there other necessary conditions on the extended boundary word EW? For example, we know that each pair of positive vertices may be involved in at most two ab-saddles one opposite to the other, or just in one saddle. Therefore each letter must appear at most once, also its inverse may appear, but if so exactly once. A more efficient algorithm would clearly be possible if we knew a better set of

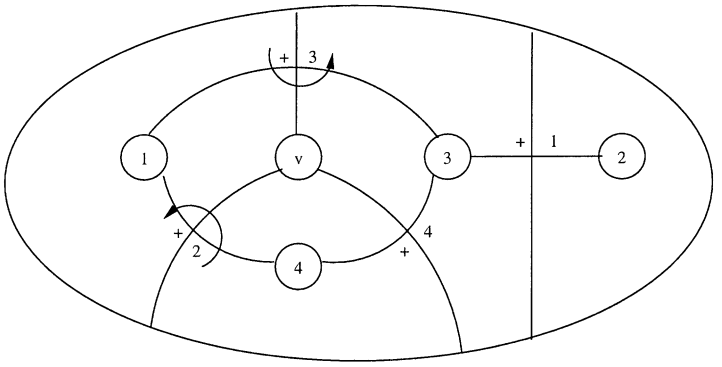

Fig. 7. A non embeddable disc.

conditions that would allow us to rule out certain discs on the basis of the associated boundary words.

3. Positive discs, good words and how to reduce redundancy

In a disc without negative vertices, condition (3) of Theorem 4 is vacuous, because there are no negative vertices, and so there are no b-arcs. This means that the embeddability test is considerably simpler for positive words than for arbitrary words, because the difficult part is the test for b-arcs.

Proposition 2. The word W associated to a positive disc with $(n, 0)$ vertices has the following properties:

1. Its length in the band generators is $n - 1$;
2. The induced permutation $\rho(W)$ is an n-cycle.

Moreover any word with these properties corresponds to the boundary word of an embeddable positive disc. Also, this word encodes all information about the disc.

Proof. The first part is a corollary of Proposition 1. The second part is known, because the braids in question are in fact the 'Stallings braids', which form a proper subset of the braids whose closure is the unknot (see eg [22]). For the last sentence, as we have seen in the preceding section, the extended word alone carries all information except the order of negative vertices in the vertex string. But a positive disc has no negative vertices. □

Definition 3. We call any word satisfying the two conditions of the preceding Proposition a **good word**.

As we anticipated in the remark at the end of the previous section, a good word completely determines the embedding of its disc. Indeed, a description of this embedding is computed by our GAP procedure GenerateDiscBoundary(n, W).

To follow the program of Theorem 6 we first need to list all the positive discs, that correspond to all the positive good words. But since we are interested in

conjugacy classes of braids, we can reduce a lot of redundancy at this stage by some easy conjugations. The following definition concerns all embeddable discs (not only the positive ones).

Definition 4. We say that two embeddable foliated discs are **equivalent** if they only differ by a cyclic permutation of the names of the vertices or saddles.

It is clear that two equivalent discs are isotopic. That is, not only are their boundary words equivalent as cyclic words, but in fact the entire singular foliation is the same, up to a cyclic permutation of the 'names' of the vertices and saddles.

Proposition 3. The (extended) boundary words W, W' of two equivalent discs D, D' only differ by some of the following easy conjugations:

1. conjugations by powers of δ, where $\delta = (n, n-1)(n-1, n-2)\dots(3,2)(2,1)$, and $\delta^{-1}(i,j)\delta = (i+1, j+1) \pmod{(n,n)}$ for all band generators (i,j) (see [3]);

2. conjugations by initial or final subwords;

Proof. Conjugations of the first kind correspond to cycling the names of vertices along the braid axis; conjugations of the second kind correspond to cycling the names of saddles about the braid axis. □

We call them 'easy conjugations' because it is very easy and inexpensive to perform them on a computer (see the Appendix).

So for a given n we will list all good words up to easy conjugations and inversions: see Proposition 7. Our GAP function EnumeratePositiveGoodWords(P) enumerates one representative for each orbit of the action of the group $G = S_P \times S_{P-1}$ on the set of positive words of B_P with length $P-1$, where S_P acts on the P indices and S_{P-1} acts on the letters of the word, discarding those which have not the required permutation property.

Once we have all positive good words up to easy conjugations, we can list all positive good words up to inversion by choosing $1, 2, \dots \lfloor \frac{P-1}{2} \rfloor$ letters of each word to become negative. This is done by another easy procedure, EnumerateGoodWords(P). Remember that words with different exponent sum are surely non conjugate.

For instance, the result of EnumerateGoodWords(4) is a list of 32 good words of exponent sum 3 or 1. We will get 32 other good words of exponent sum -3 and -1 by inversion. Notice that among these 64 good words, some represent the same braid, because it is possible to apply some relations: for instance $(2,1)\overline{(3,1)}(4,3) = (2,1)(4,1)\overline{(3,1)}$, and both words appear in the list; but notice that $(2,1)(3,1)(4,3) \neq (2,1)(4,1)(3,1)$, also both appearing in the list. So we cannot reduce the list also by relations, both because relations are different on words with different signs, and they are not easy to be performed by the computer.

S. J. Lee has reduced the 32 good words up to conjugation: there are 3 conjugacy classes with exponent sum 3, with representatives $(2,1)(3,1)(4,1)$, $(2,1)(3,1)(4,2)$

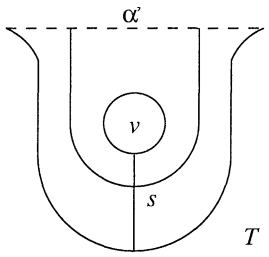

Fig. 8. An *ab*-tile.

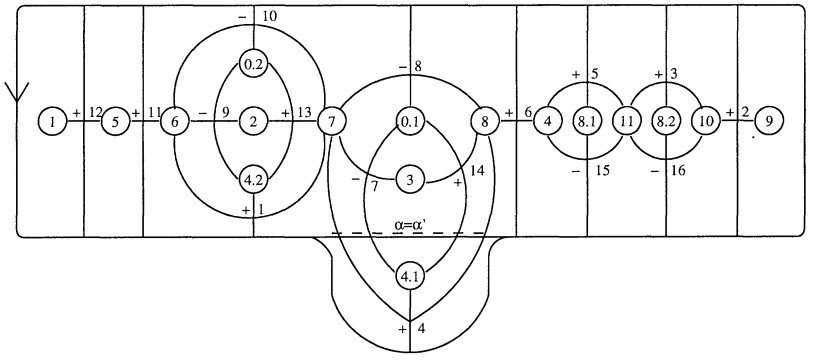

Fig. 9. The foliated disc obtained by insertion of an *ab*-tile from the disc of Figure 6.

and $(2,1)(4,2)(3,1)$. There are four conjugacy classes with exponent sum 1, with representatives $\overline{(2,1)}(3,1)(4,1)$, $\overline{(2,1)}(4,1)(3,1)$, $\overline{(2,1)}(3,1)(4,2)$ and $(2,1)(4,2)\overline{(3,1)}$.

Question 3. Are easy conjugations and defining relations in the braid group the only possible moves between conjugate positive EW's of the same length? A better understanding of this issue would lead to a more efficient method of listing the positive words which we need to test. As will be seen, any redundancies which can be eliminated at this stage of the algorithm will lead to major savings at subsequent stages, enabling us to collect better data.

4. How to insert new *ab*-tiles in a given disc

The process of inserting an *ab*-tile in a given foliated disc D is explained well in [2]. The idea is to take another small disc T (see Figure 8), with a negative vertex v and a saddle s inside it, and a distinguished arc α' on its boundary ∂T, such that the four branches of s end one in v, two in α' and the fourth in $\partial T \setminus \alpha'$. Then choose an **insertion arc** α along the boundary ∂D and attach T along α by identifying α with α' (see Figure 9), and continue branches of saddles of D ending in α inside T till they arrive at v, and continue the two branches of s ending in α' inside D till they arrive at two specified positive vertices of D. Get a new foliated disc D' in such a way that this is still embeddable.

We know the conditions for embeddability from Theorem 4, so we can impose conditions on the choice of the insertion arc α, on the position of v in the new vertex string of D', and the position of s in the new saddle list of D', so that at least the first two conditions of embeddability are satisfied.

We remark that there is another reduction of redundancy studied in [2], which is the requirement that all b-arcs be essential: We say that a foliated disc is **essential** if a negative vertex is never attached by a saddle to a positive vertex which is adjacent to it in the vertex string. If it was inessential, we could reduce the foliation of the disc by eliminating that negative vertex and the saddle attaching it to an adjacent positive vertex, without altering the embeddability of the disc and the boundary braid. So from now on we will discard all inessential discs. Since the test for essentiality is very easy, this is a very inexpensive way to reduce redundancy in our list. After we have tested the first two embeddability conditions and eliminated inessential b-arcs we will run the third text for embeddability in Theorem 4.

In order to keep track of the information necessary for a possibly essential embeddable insertion, we have invented **another combinatorial description of the disc** D, also suitable for our implementation. We will code our disc D by two sets: the cyclically ordered set of boundary points of saddles, read counterclockwise along ∂D, and the set of bb-saddles. The reader might like to compare what follows with Figure 6. Each aa-saddle has two points on the boundary ∂D, that we will call **points of type Q**; each ab-saddle has one point on ∂D that we will call **points of type R**; and each bb-saddle lies entirely in the interior of D, with no point on ∂D.

With each boundary point we will associate: the sign of the attached saddle (overline points corresponding to negative saddles); the ordered list of two or three vertices to which the saddle is attached, and the ordinal number of the saddle.

The double index of $Q_{i,j}$ (or $\overline{Q}_{i,j}$) is such that the cyclic counterclockwise order about the saddle of the boundary point and the two vertices is i, Q, j.

The triple index of $R_{i,v,j}$ (or $\overline{R}_{i,v,j}$) is such that the cyclic clockwise order of this point and the three vertices around the saddle is R, i, v, j (in particular, the central index is the negative vertex).

We call **initial and final vertex** of a boundary point respectively the first and the last positive vertex, as they occur as indices of the point. In the sequence read along the boundary, two consecutive points always have the final vertex of the preceding point equal to the initial vertex of the following point.

Each bb-tile can be coded by: the sign of the saddle, the ordered sequence of the four vertices around it, and its ordinal number.

These combinatorial data are clearly in bijective correspondence with the set of data given in Section 2, so they are sufficient to draw the foliated disc and to read the extended boundary word (cf [2]).

For instance, the disc of Section 2 is described by (cf Figure 6)

$$\partial D = \{Q_{1,5}(11), Q_{5,6}(10), R_{6,4.1,7}(1), \overline{R}_{7,0.1,3}(6), R_{3,0.1,8}(13), Q_{8,4}(5),$$

$$\overline{R}_{4,8.1,11}(14), \overline{R}_{11,8.2,10}(15), Q_{10,9}(2), Q_{9,10}(2), R_{10,8.2,11}(3), R_{11,8.1,4}(4),$$

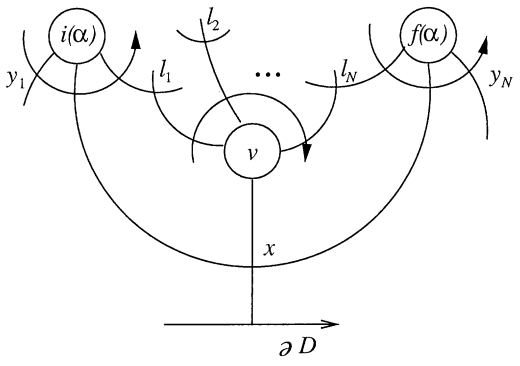

Fig. 10. How to find the possible level of the new saddle.

$$Q_{4,8}(5), \overline{R}_{8,0.1,7}(7), \overline{R}_{7,0.2,6}(9), Q_{6,5}(10), Q_{5,1}(11)\};$$
$$bb\text{-saddles} = \{\overline{[6, 0.2, 2, 4.1]}(8), [7, 0.2, 2, 4.1](12)\}.$$

We code an insertion arc α by listing all the consecutive points of type **Q** and **R** contained in α. The position of the new inserted negative vertex v can be found as follows:

- for each point $Q_{i,j}$ or $R_{i,u,j}$ in α: if $j - i \leq 2(\text{mod } n)$, then the insertion is inessential; otherwise we get $i + 1 < v < j - 1$;
- for each point $\overline{Q}_{i,j}$ or $\overline{R}_{i,u,j}$ in α: if $i - j \leq 2(\text{mod } n)$, then the insertion is inessential; otherwise we get $j + 1 < v < i - 1$.

These conditions correspond to the cyclic order of vertices around saddles. This means also that if α contains more than one point, we have to intersect conditions coming from different points: if the intersection is empty, the insertion is not embeddable. We can list all possible intervals of insertion in order of increasing length (that is the number of singular boundary points), giving for each of them the possible essential position of the new negative vertex. If some of the two preceding conditions eliminate some arc α of length k, then all arcs with length greater than k and containing α are inessential or not embeddable for the same reason. The complete list is done by our GAP procedure GetInsertionArcs$(n, \partial D)$.

In our example (see Figure 6), an insertion arc is for instance $\alpha = \{\overline{R}_{7,0.1,3}, R_{3,0.1,8}\}$, with $4 < v < 6$ from the first point and $4 < v < 7$ from the second, hence it must be $4 < v < 6$.

An insertion arc α has an **initial** and a **final vertex** $(i(\alpha), f(\alpha))$, respectively the initial vertex of its first point, and the final vertex of its last point. In our example they are $(7, 8)$.

To know at which level x the new saddle can be located, we need to look at levels of points of α and to consider the cyclic order of all saddles around the new vertex, and around the two positive vertices $i(\alpha), f(\alpha)$ (see Figure 10).

Suppose $\alpha = \{P_1(l_1), P_2(l_2), \dots P_N(l_N)\}$, where each P_j can be either of $\mathbf{Q}$ or of $\mathbf{R}$ type, either positive or negative, and each l_j indicates its level. Then we must have $l_1 < l_2 < \cdots < l_N < x$ (in cyclic order). Now we have to look for the point on ∂D immediately preceding P_1 and containing the index $i(\alpha)$: suppose y_1 is its level: then we must have $y_1 < x < l_1$. Also we have to look for the point on ∂D immediately following P_N and containing the index $f(\alpha)$: suppose y_N is its level: then we must have $l_N < x < y_N$.

In our example: $6 < 13 < x$, that means $13 < x < 6$, is the first requirement; $l_1 = 1$ and $l_N = 5$, therefore $1 < x < 6$ and $13 < x < 5$. Intersecting these cyclic intervals we get $1 < x < 5$.

The sign of the new saddle only depends on the relative position of $i(\alpha), v, f(\alpha)$:

- if $f(\alpha) < v < i(\alpha)$ then s is positive;
- if $i(\alpha) < v < f(\alpha)$ then s is negative;

therefore the possible range of v obtained before can be divided in two parts, giving different signs for s. Our GAP procedure GetSaddles$(n, P + N - 1, \partial D)$ gives all these results.

In our example, $f(\alpha) = 8 < 4 < v < 6 < 7 = i(\alpha)$ hence s is positive.

At this stage, before proceeding with the expensive b-arcs test, we can perform an easy **permutation test**: when we do an insertion, it is easy to see how the corresponding extended word changes: a new saddle $[\pm 1, [i(\alpha), v, f(\alpha))], x]$ is inserted, this corresponds to inserting a new letter $(i(\alpha), f(\alpha))^{\pm 1}$ (or $(f(\alpha), i(\alpha))^{\pm 1}$) at the same level of EW: we get a longer word, which still must satisfy conditions given in Proposition 1 for an extended word. So we can try all possible combinations of $[\pm 1, [i(\alpha), v, f(\alpha))], x]$ and check the corresponding permutation, to discard the impossible ones.

For instance our possible insertions for the chosen arc α are $[+1, [7, v, 8], x]$, with x ranging between 1 and 5. So the new word might be one of the following:

$$\begin{aligned}
EW(D_1) &= (7,6)(\mathbf{8,7})(\mathbf{10,9})(\mathbf{11,10})(\mathbf{11},4)(\mathbf{8},4)\overline{(\mathbf{7,3})(\mathbf{8,7})(\mathbf{6,2})}\cdot \\
&\quad \cdot\overline{(7,6)}(6,5)(5,1)(7,2)(8,3)\overline{(11,4)(11,10)}; \\
EW(D_2) &= (7,6)(10,9)(\mathbf{8,7})(\mathbf{11,10})(\mathbf{11},4)(\mathbf{8},4)\overline{(\mathbf{7,3})(\mathbf{8,7})(\mathbf{6,2})}\cdot \\
&\quad \cdot\overline{(7,6)}(6,5)(5,1)(7,2)(8,3)\overline{(11,4)(11,10)}; \\
EW(D_3) &= (7,6)(10,9)(11,10)(\mathbf{8,7})(\mathbf{11},4)(\mathbf{8},4)\overline{(\mathbf{7,3})(\mathbf{8,7})(\mathbf{6,2})}\cdot \\
&\quad \cdot\overline{(7,6)}(6,5)(5,1)(7,2)(8,3)\overline{(11,4)(11,10)}; \\
EW(D_4) &= (7,6)(10,9)(11,10)(11,4)(\mathbf{8,7})(\mathbf{8},4)\overline{(\mathbf{7,3})(\mathbf{8,7})(\mathbf{6,2})}\cdot \\
&\quad \cdot\overline{(7,6)}(6,5)(5,1)(7,2)(8,3)\overline{(11,4)(11,10)}.
\end{aligned}$$

For them we find the following permutations:

$$\rho(EW(D_j)) = (1,5,6,10,9)(2)(3)(4)(7)(8)(11)$$

(the same for all, since $(8,7)$ commutes with second, third and fourth letters), which satisfies all conditions required by Proposition 1.

Now we have to perform the test for b-arcs. For this we need both ∂D and the set of bb-saddles. Also, we first need to see **how the code changes after an insertion**.

- if $i < v < i+1$ and in the same interval there are other existing negative vertices, we have to decide (by the embeddability test) the exact order of them in this interval, and rename old vertices in it if necessary;
- if $y - 1 < x < y$, put $x = y$ and for any saddle which was at level $z \geq y$ put it at level $z + 1$;
- substitute α by $R^{sign(s)}_{i(\alpha),v,f(\alpha)}$;
- for each point $Q^{\varepsilon}_{i,j}$ in α, its other corresponding $Q^{\varepsilon}_{j,i}$ becomes $R^{\varepsilon}_{j,v,i}$;
- each point $R^{\varepsilon}_{i,u,j}$ in α becomes a bb-saddle $[\varepsilon,[i,u,j,v]]$.

If for instance we make the insertion of $[+1,[7,4.1,8],4]$ on the chosen α, the description of the new disc is (see Figure 9):

$$\partial D = \{Q_{1,5}(12), Q_{5,6}(11), R_{6,4.2,7}(1), R_{7,4.1,8}(4), Q_{8,4}(6),$$

$$\overline{R}_{4,8.1,11}(15), \overline{R}_{11,8.2,10}(16), Q_{10,9}(2), Q_{9,10}(2), R_{10,8.2,11}(3), R_{11,8.1,4}(5),$$

$$Q_{4,8}(6), \overline{R}_{8,0.1,7}(8), \overline{R}_{7,0.2,6}(10), Q_{6,5}(11), Q_{5,1}(12)\};$$

$$bb\text{-saddles} = \{\overline{[0.1,3,4.1,7]}(7), \overline{[6,0.2,2,4.2]}(9),$$

$$[7,0.2,2,4.1](13), [0.1,3,4.1,8](14)\}.$$

How to read the b-arcs: if a negative vertex v (yet existing or newly inserted) is attached to only two saddles (for the new vertex, this corresponds to an insertion arc of length 1), they (see Figure 11) will surely be one positive (say at level y_+) and one negative saddle (say at level y_-), and they will be connected to the same two positive vertices, say i and j. If $i < v < j$, then in the interval (y_+, y_-) we have the b-arc $b(j,v)$ and in the interval (y_-, y_+) we have the b-arc $b(i,v)$.

If a negative vertex v (yet existing or newly inserted) is attached to more than two saddles (for the new vertex, this corresponds to an insertion arc of length greater than or equal to 2), we have to list these saddles in cyclic order about v (in our example, they are $[7,4.1,8](4)$, $\overline{[7,0.1,3,4.1]}(7)$, $[3,0.1,8,4.1](14)$); between any two consecutive of them, find the b-arc between v and the only other positive vertex

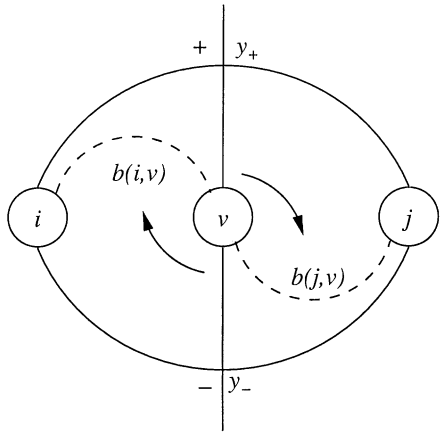

Fig. 11. How to find the b-arcs about a negative vertex.

which is in common for the two saddles (in our example: in $(4,7)$ have $b(7,4.1)$; in $(7,14)$ have $b(3,4.1)$; in $(14,4)$ have $b(8,4.1)$).

As explained in [2], we have to compose an array with: in the first column the $P+N-1$ intervals $(k,k+1)$ between two consecutive saddles; in the following N columns, the b-arcs for each negative vertex in the corresponding intervals; in the last column, the gb-arcs (they are as many as the aa-saddles). For instance, for our new disc of Figure 9 we see the array of Figure 12.

Remark: When an aa-saddle occurs, no b-arc is changed; when an ab-saddle occurs, only changes the b-arc connected to the negative vertex involved in the ab-saddle; when a bb-saddle occurs, only change the two b-arcs connected to the negative vertices involved in the bb-saddle.

It is necessary that along each row of the array the arcs which appear do not interlock with each other. All these checks are performed by our GAP procedure Embeddable$(n, P+N-1, \partial D, bb\text{-saddles})$. Finally, our GAP procedure InsertVertices$(n, P+N-1, \partial D, bb\text{-saddles}, \text{depth})$ recursively tries all possible insertions of ab-tiles and returns all the resulting new embeddable essential tiled disc.

5. How to get the boundary word. Changes in foliation.

To get the boundary word BW we have to eliminate from EW the N strands corresponding to the N unlinked circles about each negative vertex. In this way, from each letter of EW we will get a new letter or subword of the boundary braid BW, which is a braid of B_{P-N}.

To do so, let us introduce some useful braids (called *descending cycles* in [3]):

Definition 5. When $p > q$, call $\delta_{p,q}$ the braid $(p,p-1)(p-1,p-2)\cdots(q+1,q)$.

$(1,2)$	$b(3,0.1)$	$b(2,0.2)$	$b(8,4.1)$	$b(6,4.2)$	$b(4,8.1)$	$b(11,8.2)$	$gb(9,10)$
$(2,3)$	$b(3,0.1)$	$b(2,0.2)$	$b(8,4.1)$	$b(6,4.2)$	$b(4,8.1)$	$b(11,8.2)$	
$(3,4)$	$b(3,0.1)$	$b(2,0.2)$	$b(8,4.1)$	$b(6,4.2)$	$b(4,8.1)$	$b(10,8.2)$	
$(4,5)$	$b(3,0.1)$	$b(2,0.2)$	$b(7,4.1)$	$b(6,4.2)$	$b(4,8.1)$	$b(10,8.2)$	
$(5,6)$	$b(3,0.1)$	$b(2,0.2)$	$b(7,4.1)$	$b(6,4.2)$	$b(11,8.1)$	$b(10,8.2)$	$gb(4,8)$
$(6,7)$	$b(3,0.1)$	$b(2,0.2)$	$b(7,4.1)$	$b(6,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(7,8)$	$b(7,0.1)$	$b(2,0.2)$	$b(3,4.1)$	$b(6,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(8,9)$	$b(8,0.1)$	$b(2,0.2)$	$b(3,4.1)$	$b(6,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(9,10)$	$b(8,0.1)$	$b(6,0.2)$	$b(3,4.1)$	$b(2,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(10,11)$	$b(8,0.1)$	$b(7,0.2)$	$b(3,4.1)$	$b(2,4.2)$	$b(11,8.1)$	$b(10,8.2)$	$gb(5,6)$
$(11,12)$	$b(8,0.1)$	$b(7,0.2)$	$b(3,4.1)$	$b(2,4.2)$	$b(11,8.1)$	$b(10,8.2)$	$gb(1,5)$
$(12,13)$	$b(8,0.1)$	$b(7,0.2)$	$b(3,4.1)$	$b(2,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(13,14)$	$b(8,0.1)$	$b(2,0.2)$	$b(3,4.1)$	$b(7,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(14,15)$	$b(3,0.1)$	$b(2,0.2)$	$b(8,4.1)$	$b(7,4.2)$	$b(11,8.1)$	$b(10,8.2)$	
$(15,16)$	$b(3,0.1)$	$b(2,0.2)$	$b(8,4.1)$	$b(7,4.2)$	$b(4,8.1)$	$b(10,8.2)$	
$(16,1)$	$b(3,0.1)$	$b(2,0.2)$	$b(8,4.1)$	$b(7,4.2)$	$b(4,8.1)$	$b(11,8.2)$	

Fig. 12. Table of b- and gb-arcs.

Notice that $\delta_{p,p-1} = (p, p-1) = \sigma_{p-1}$. The associated permutation is $\rho(\delta_{p,q}) = (q, q+1, \ldots, p)$.

The strands to be eliminated are numbered, at the beginning of EW, by the N 1-cycles of $\rho(EW)$. At each letter of EW their numbers might change, as can be seen for instance in Figure 13. The rule is: suppose $L_k = \{i_1, i_2, \ldots i_N\}_k$ is the list of levels of strands to be eliminated just before the k^{th} letter of EW, and this letter is $(h,j)^\varepsilon$ (see Figure 14):

1. if both h, j are not in L_k, and h', j' are the numbers of elements of L_k which are less than h, j respectively; then get $(h-h', j-j')^\varepsilon$; $L_{k+1} = L_k$;
2. if $j \in L_k$ and h is not in L_k: if $h > j+1$, then get $\delta_{h-h',j-j'}$; if $h = j+1$, then get the empty word e; $L_{k+1} = (L_k \setminus \{j\}) \cup \{h\}$;
3. if $h \in L_k$ and j is not in L_k: if $h > j+1$, then get $\delta^{-1}_{h-1-h',j-j'}$; if $h = j+1$, then get e; $L_{k+1} = (L_k \setminus \{h\}) \cup \{j\}$;
4. if both $h, j \in L_k$, then get e; $L_{k+1} = L_k$.

In our example we get

$$BW = e(6,5)e\delta_{6,2}(5,2)\delta_{4,2}eee(5,4)(4,1)e\delta^{-1}_{5,2}\delta^{-1}_{6,2}e =$$

$$= (6,5)\delta_{6,2}(5,2)\delta_{4,2}(5,4)(4,1)\delta^{-1}_{5,2}\delta^{-1}_{6,2}.$$

Our GAP procedure BoundaryBraid(EW, P, N) computes the boundary braid of a disc with extended word EW and (P, N) vertices.

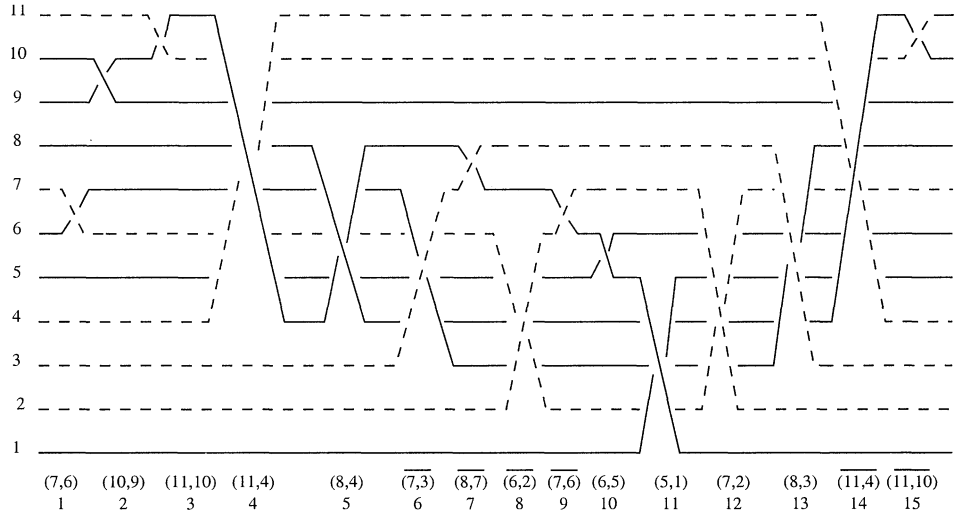

Fig. 13. The braid EW. The dashed strands must be removed to find the boundary braid.

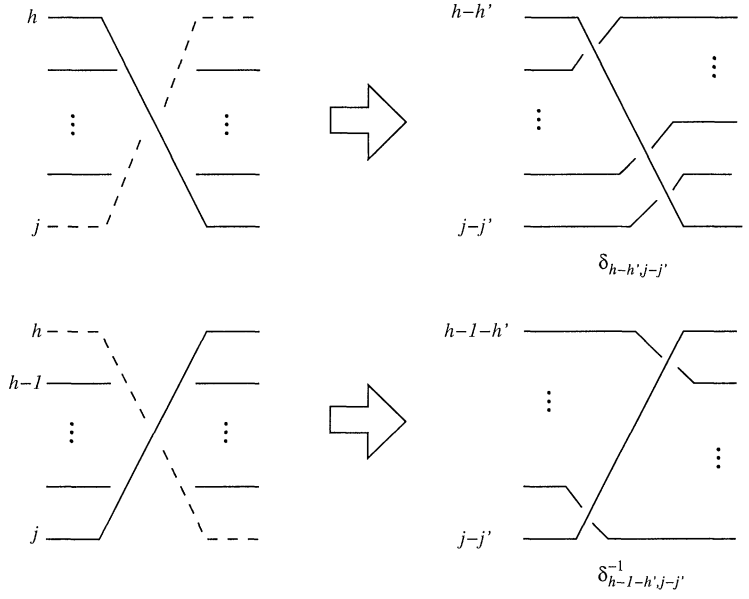

Fig. 14. The braids δ's obtained from a generator by removing one strand.

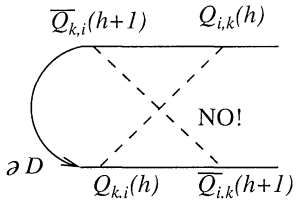

Fig. 15. Two impossible *aa*-saddles.

In what follows we explain how the defining relations and inversion in the braid group, performed on the extended word, affect the topology of the foliated disc and the corresponding boundary word.

Question 4. How do these changes in foliation affect the possibility of inserting *ab*-tiles?

To describe what happens let us first notice some properties of the $\delta_{i,j}$'s.

Proposition 4. 1. $\delta_{i,j}\delta_{h,k} = \delta_{h,k}\delta_{i,j}$ if $i > j > h > k$;

2. $\delta_{i,j}\delta_{h,k} = \delta_{h-1,k-1}\delta_{i,j}$ if $i > h > k > j$;

3. $(i,j)\delta_{h,k} = \delta_{h,k}(i+1, j+1)$ if $h > i > j > k$;

In what follows, when we say 'letter' we mean letters in band generators or δ's.

The easiest 'relation' we want to describe is free reduction $(i,k)\overline{(i,k)}$.

Proposition 5. Let EW be the extended word of an embeddable disc D with (P,N) vertices. If EW is reducible (i.e. it has two consecutive letters, one of which is the inverse of the other), then the reduced word EW' is the extended word of another embeddable disc D' with $(P-1, N-1)$ vertices. Moreover, the two boundary braids BW, BW' define the same braid.

Proof. Suppose that $w_h w_{h+1} = (i,k)\overline{(i,k)}$. Then, along ∂D we see either the two pairs of consecutive points $Q_{i,k}(h)\overline{Q}_{k,i}(h+1)$ and $Q_{k,i}(h)\overline{Q}_{i,k}(h+1)$, or one of the pairs $R_{i,v,k}(h)\overline{R}_{k,v,i}(h+1)$ or $R_{k,v,i}(h)\overline{R}_{i,v,k}(h+1)$. The first case is impossible (see Figure 15), since the two saddles would cross each other. In the second case, the positive vertex k (resp. i) is isolated from any other *aa*- or *ab*-saddle (see Figure 16); there cannot be other saddles involving k, because the two points on the boundary are consecutive, and no other *bb*-saddle can occur among the two vertices, to respect embeddability (see Figure 17). So we can reduce EW by deleting the two inverse letters, to get a braid word in B_{P-1} representing a disc D' (see Figure 16) which differs from D by not having the two corresponding saddles, the vertices k (resp. i) and v. The eliminated positive vertex, because of its position in D, is one of those in the one-cycles of the permutation. The new word EW' has length $(P-1)+(N-1)-1$ and one less one-cycle, so it still has a good permutation. We

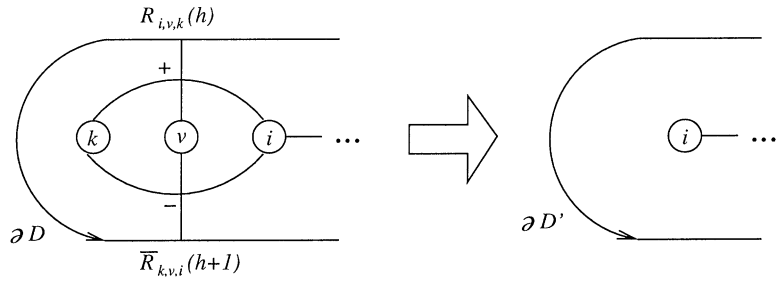

Fig. 16. Elimination of two adjacent inverse saddles.

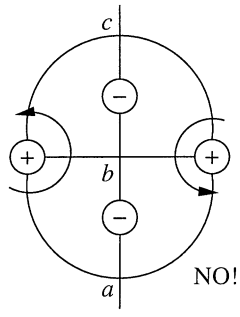

Fig. 17. An impossible *bb*-saddle.

have eliminated two ab-saddles from D, so we have eliminated some b-arcs, therefore since D was embeddable, so is D'.

The two boundary braids live in the same braid group, since $P - N = P - 1 - (N-1)$. Moreover the two consecutive inverse letters of EW go into two consecutive inverse letters of BW (maybe e), so $BW = BW'$ as braid elements. □

Remark: The inverse operation, ie the insertion in an extended word of a pair of inverse letter, is not always admissible: first of all, one of the two indices of the inserted letters must be new, or we have to add one to all indices after this. Even so, the insertion might produce an inessential or non embeddable disc: for instance given $EW = (6,5)(5,4)\overline{(4,2)}(3,1)(5,3)$, if we insert the pair $(7,2)\overline{(7,2)}$ between the second and the third letter, we get an inessential disc. If we insert in the same position the pair $(7,3)\overline{(7,3)}$, we get an impossible disc.

Proposition 6. Let EW be the extended word of an embeddable disc D. If in EW two consecutive letters commute, then the related word EW' is the extended word of another embeddable disc D' such that the two boundary braids BW, BW' are the same braid.

Proof. Suppose that $w_m w_{m+1} = (i,j)^{\varepsilon}(h,k)^{\eta}$, with $\varepsilon, \eta \in \{\pm 1\}$. Then the two corresponding saddles cannot have any common vertex. For, if there is a common vertex it is negative. But then it is impossible to foliate regularly the region about

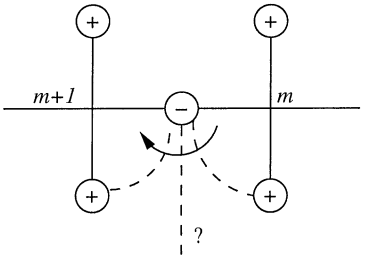

Fig. 18. Two impossible ab-saddles.

this negative vertex which is bounded by the two saddles, see Figure 18.

Therefore, exchanging the order of the two saddles does not affect the cyclic order of saddles about vertices, nor of vertices about saddles. In the interval $(m-1, m+2)$ only these two saddles occur, so only their b- or gb-arcs change their occurrence: but since they do not share any common vertex, and all the other arcs remain unchanged, this commutation does not affect the b-arc test, so D' is still embeddable.

Clearly the only change in BW is the commutation of the two corresponding letters. This commutation leads to a different braid word representing the same braid. □

Proposition 7. Let EW be the extended word of an essential embeddable disc D. Then EW^{-1} is the extended boundary word of an essential embeddable disc $\overline{D}$, which can be drawn as follows: look at D from its negative side, let all names of the (P, N) vertices unchanged, change sign to all saddles, and change numbers of saddles by the reversing permutation $(1, P+N-1)(2, P+N-2)\ldots$

Proof. For this proof, we use results of section 6. D is essential and embeddable if and only if it has an essential and embeddable H_θ-sequence. Consider the sequence read from it in the reverse order: so each ab- or bb-saddle changes its sign; assign opposite sign also to each aa-saddle. The resulting sequence is clearly essential and embeddable as the previous one, and it corresponds to the inverse extended boundary word: it has inverse induced permutation, hence a good one again. If we draw the disc starting from EW^{-1} we get saddles in the reverse order. □

An example is shown in Figure 26.

Proposition 8. Let EW be the extended word of an embeddable disc D. If in EW we can perform a relation between two consecutive letters sharing an index, then the related word EW' is the extended word of another embeddable disc D' such that the two boundary braids BW, BW' are the same braid.

Proof. The configurations on D, D' of the possible relations are given in Figure 19. If all three saddles are of aa-type, so that none of the points A, B, C is a negative vertex, then the embeddability is unchanged: in fact, in the interval $(m-1, m+2)$ only these two aa-saddles occur, and the two corresponding gb-arcs change in such

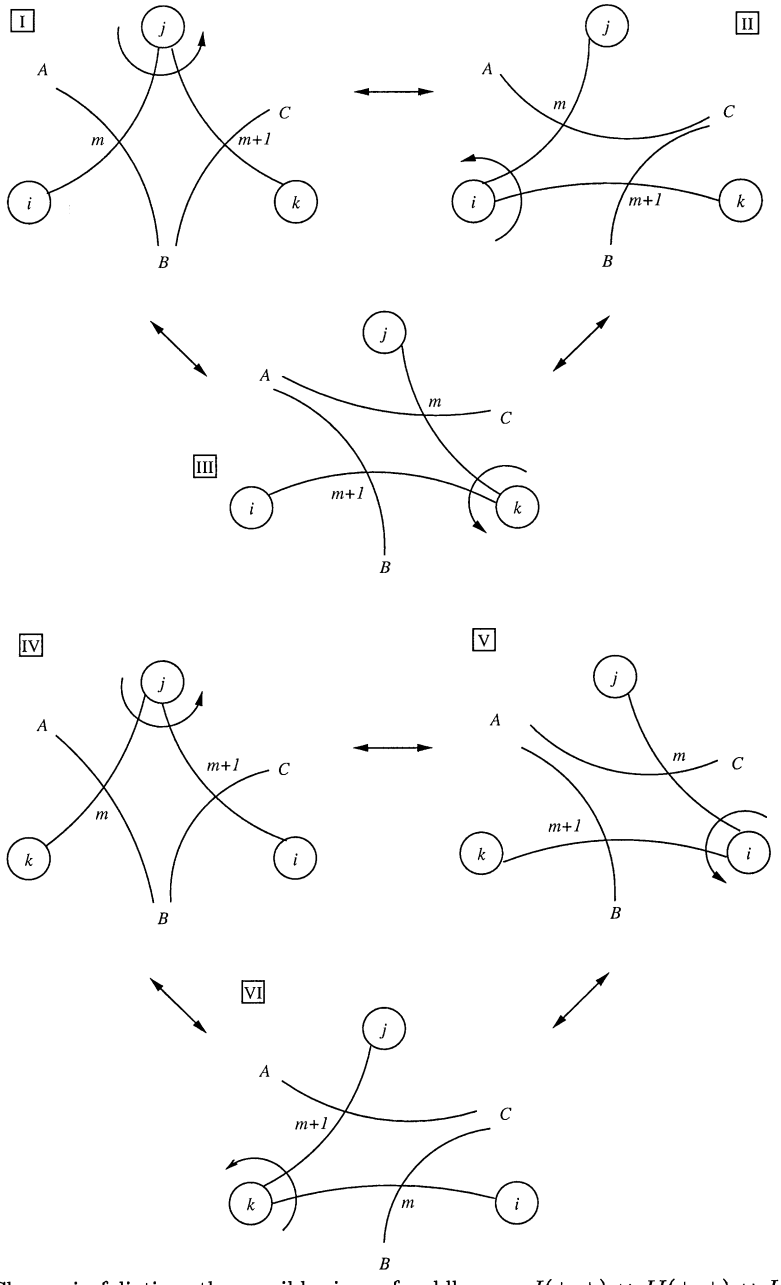

Fig. 19. Change in foliation: the possible signs of saddles are: $I(+,+) \leftrightarrow II(+,+) \leftrightarrow III(+,+)$; $I(+,-) \leftrightarrow II(-,+)$; $II(+,-) \leftrightarrow III(-,+)$; $III(+,-) \leftrightarrow I(-,+)$; $IV(-,-) \leftrightarrow V(-,-) \leftrightarrow VI(-,-)$; $IV(+,-) \leftrightarrow V(-,+)$; $V(+,-) \leftrightarrow VI(-,+)$; $VI(+,-) \leftrightarrow IV(-,+)$.

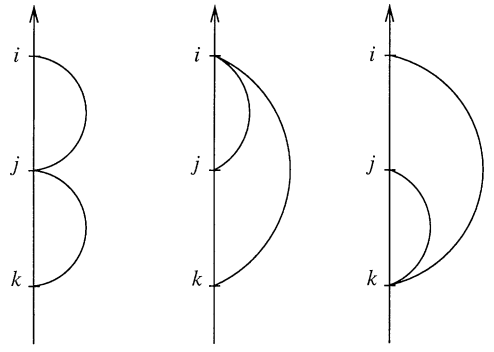

Fig. 20. Relations that do not change the possibility of drawing b-arcs.

a way that the possibility of drawing b-arcs in their complement does not change (see Figure 20).

If some of the points A, B, C are negative vertices, we also have to take account of the change of b-arcs. In this case too, by examining all possible cases, we see that the possibility of drawing the other b-arcs is not affected by the relation.

The change in the boundary word is a corresponding relation, possibly between δ's or between some δ and some band generator, but always giving a related boundary word representing the same braid element. □

Remark: Relations do not affect embeddability, but they might affect essentiality. Moreover, it is true that they change the topology of the embeddable disc, so that, for instance, they can affect the possibility of inserting an ab-tile. In fact, they always change the possible insertion arcs. This leads to the following:

Question 5. We phrase this question as a conjecture: The list of good words to be used to find new words by insertions of ab-tiles cannot be reduced by performing relations of the braid group.

Here is an example:

$$W_1 = (65)(54)\overline{(42)}(31)(53); W_2 = (65)\overline{(52)}(54)(31)(53);$$

these two words represent the same braid, since they only differ by a relation. They are both good words, therefore they represent embeddable positive discs. But when we examine them, we find that in D_1 we can perform two embeddable essential insertions:

1. along $\overline{Q}_{2,4}(3)$ with $6 < v < 1$ and positive saddle at level $4 < x < 5$;
2. along $Q_{5,3}(5)$ with $6 < v < 1$ and negative saddle at level $2 < x < 3$.

The extended and boundary words of the discs we get from these insertions are respectively:

1. $E_1 = (65)(54)\overline{(42)}(31)(42)(53)$ and $B_1 = (54)(43)(32)(31)\overline{(32)}(42)$;
2. $E_2 = (65)(54)\overline{(53)(42)}(31)(53)$ and $B_2 = (54)(43)(43)\overline{(42)}(31)\overline{(43)}$.

Notice that these two 5-braids are not conjugate, since they have different exponent sum.

But if we look at W_2, we find that only one essential embeddable insertion is possible, namely $\{\overline{Q}_{25}(2), Q_{54}(3)\}$, with $6 < v < 1$ and positive saddle at level $4 < x < 5$, from which we get the extended and the boundary word

$$E_3 = (65)\overline{(52)}(54)(31)(42)(53),\ B_3 = (54)(43)(32)(31)\overline{(32)}(42),$$

which is the same we got from the first insertion on W_1. In fact this insertion does correpond to the other one, since the change in foliation given by the relation has substituted the arc containing $\overline{Q}_{24}$ with the arc containing now $\overline{Q}_{25}, Q_{54}$.

Remark: We might find the other 5-braid from a different good word, related to W_1 in some other way. Or if we list our discs using the method of H_θ-sequences (see section 6) instead of insertions of ab-tiles, we have not to be worried by these changes in foliation.

Change in the code after a relation (for good words):

Suppose W is a good word and we perform a relation between two consecutive letters $w_h w_{h+1}$. Then the code for W changes as follows:

Commutation of non interlocking pairs:

$$(ij)^\varepsilon (kl)^\eta \longleftrightarrow (kl)^\eta (ij)^\varepsilon$$

(here $\varepsilon, \eta \in \{\pm 1\}$).

Change in the code: exchange levels $h \longleftrightarrow h+1$ of the four Q points.

Relation between two positive letters with three ordered indices $n \geq i > j > k \geq 1$:

$$(ij)(jk) \longleftrightarrow (jk)(ik) \longleftrightarrow (ik)(ij) \longleftrightarrow$$

Change in the code: three arcs of the boundary change as follows:

$$Q_{ij}(h)Q_{jk}(h+1) \longleftrightarrow Q_{ik}(h+1) \longleftrightarrow Q_{ik}(h) \longleftrightarrow;$$

$$Q_{ji}(h) \longleftrightarrow Q_{jk}(h)Q_{ki}(h+1) \longleftrightarrow Q_{ji}(h+1) \longleftrightarrow;$$

$$Q_{kj}(h+1) \longleftrightarrow Q_{kj}(h) \longleftrightarrow Q_{ki}(h)Q_{ij}(h+1) \longleftrightarrow .$$

Relation between two negative letters with three ordered indices $n \geq i > j > k \geq 1$:

$$\overline{(jk)(ij)} \longleftrightarrow \overline{(ik)(jk)} \longleftrightarrow \overline{(ij)(ik)} \longleftrightarrow$$

Change in the code: three arcs of the boundary change as follows:

$$\overline{Q}_{jk}(h) \longleftrightarrow \overline{Q}_{jk}(h+1) \longleftrightarrow \overline{Q}_{ji}(h)\overline{Q}_{ik}(h+1) \longleftrightarrow;$$

$$\overline{Q}_{kj}(h)\overline{Q}_{ji}(h+1) \longleftrightarrow \overline{Q}_{ki}(h) \longleftrightarrow \overline{Q}_{ki}(h+1) \longleftrightarrow;$$

$$\overline{Q}_{ji}(h+1) \longleftrightarrow \overline{Q}_{jk}(h+1) \longleftrightarrow \overline{Q}_{ji}(h)\overline{Q}_{ik}(h+1) \longleftrightarrow .$$

Relation between one positive and one negative letter with three cyclically ordered indices $i > j > k$:

$$(jk)\overline{(ik)} \longleftrightarrow \overline{(ij)}(jk)$$

Change in the code: three arcs of the boundary change as follows:

$$Q_{jk}(h)\overline{Q}_{ki}(h+1) \longleftrightarrow \overline{Q}_{ji}(h);$$

$$Q_{kj}(h) \longleftrightarrow Q_{kj}(h+1);$$

$$\overline{Q}_{ik}(h+1) \longleftrightarrow \overline{Q}_{ji}(h)Q_{ik}(h+1).$$

The other relation between one positive and one negative letter with three cyclically ordered indices $i > j > k$:

$$(jk)\overline{(ij)} \longleftrightarrow \overline{(ij)}(ik)$$

Change in the code: three arcs of the boundary change as follows:

$$Q_{jk}(h) \longleftrightarrow \overline{Q}_{ji}(h)Q_{ik}(h+1);$$

$$Q_{kj}(h)\overline{Q}_{ji}(h+1) \longleftrightarrow Q_{ki}(h+1);$$

$$\overline{Q}_{ij}(h+1) \longleftrightarrow \overline{Q}_{ij}(h).$$

A similar description could be done for any extended word.

Non conjugated good words giving the same disc after an insertion:

These words can be found from an essential embeddable disc with one negative vertex, by stabilizing along one ab-tile (stabilizing is the opposite process than inserting an ab-tile).

For instance, if the negative vertex is connected just to two ab-saddles 'facing each other', of opposite sign and at different levels (eg saddles 4 and 14 in Figure 6), by stabilizing along one or the other we get words with different exponent sum, hence non conjugate.

More precisely: if two words only differ for one letter (ki) which is positive and at level s in the first word, but negative and at level t in the second, then if we insert an ab-tile with negative vertex at level j, but: on $Q_{ki}(s)$ with negative saddle at level t on the first disc, and on $\overline{Q}_{ik}(s)$ with positive saddle at level s on the second, we get the same disc.

6. The H_θ-sequence

As we have seen, the test for b-arcs in the embeddability test is very expensive. But there is a different method for testing embeddability, used repeatedly in the papers of the first author and Menasco, e.g. see the proof of Lemma 3 in [5]. Using it, we can avoid the expensive part of the embeddability test.

Look at the situation from another point of view: not at the foliation on the disc, but at the foliations on the half-planes H_θ's running about the braid axis A. In fact, almost all H_θ's have $P - N$ a-arcs and N b-arcs (all regular leaves for D), except $P + N - 1$ of them, in each of which a saddle occurs. Among two consecutive singular half-planes, all infinite regular ones are uniquely identified up to isotopy by the N b-arcs. Also, the passage through one singular half-plane is such that:

- if an aa-saddle occurs, then no b-arc changes;
- if an ab-saddle occurs: then one b-arc changes its positive vertex;
- if a bb-saddle occurs: then two b-arcs exchange their positive vertices.

Moreover, the way in which these changes occur uniquely specifies the type, the names and the sign of the ab- or bb-saddle. Only the aa-saddles remain unspecified in names and sign. In Figure 21 we show all these changes and the corresponding saddles.

Proposition 9. A half-plane H_θ with specified vertex string V with (P, N) vertices is embeddable and essential if and only if the N b-arcs do not intersect each other, and each of them does not connect two consecutive vertices. ||

Given a vertex string, it is very easy to list all possible essential and embeddable regular half-planes with these vertices.

Proposition 10 (cf [5]). There is a bijective correspondence between essential embeddable foliated discs with (P, N) vertices and H_θ-sequences of length $P + N - 1$ satisfying the following properties:

- Among two consecutive half-planes in the sequence the unique change is one corresponding to a saddle;
- The permutation associated to the resulting cycle of saddles satisfies the properties of Proposition 1;
- No saddle occurs twice; the unique saddles involving the same pair of positive vertices can be two ab-saddles of opposite sign;
- All vertices occur at least in one saddle. ||

So this is **our algorithm**:

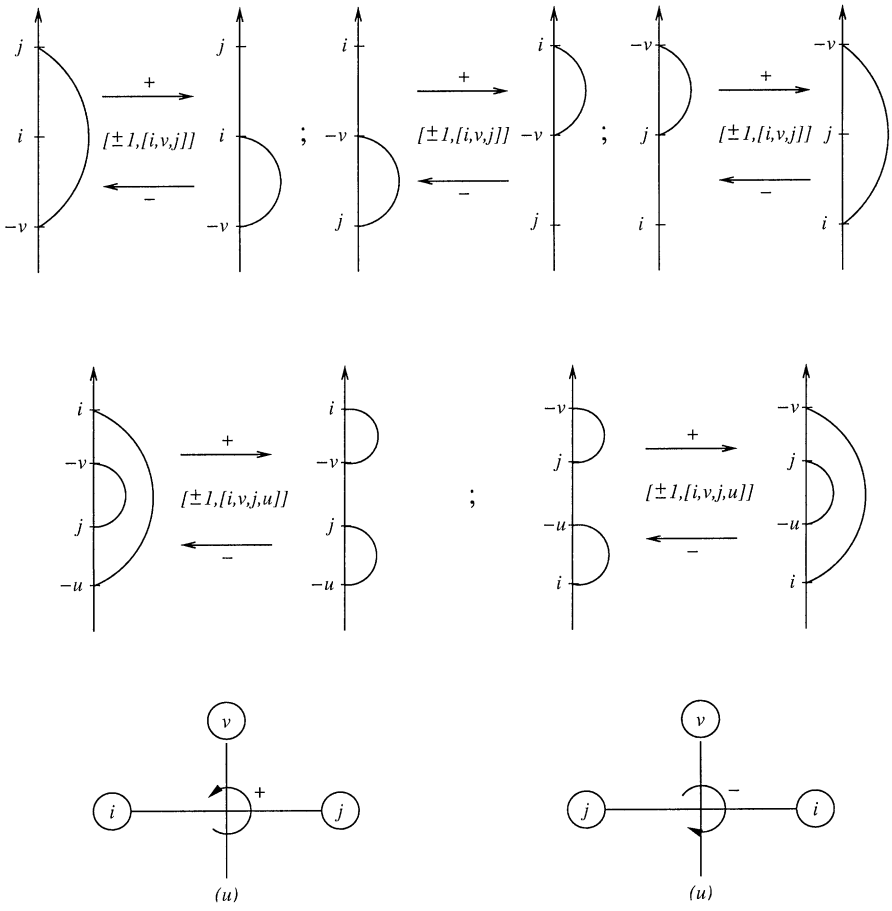

Fig. 21. The *ab*-saddles and *bb*-saddles seen on the halfplanes.

1. For each (P, N), with $P > N + 1$, list all possible vertex strings, up to cyclic order along the axis.

2. For each vertex string, list all possible embeddable essential regular half-planes;

3. Given the complete set of half-planes found for a given vertex string, list all the cyclic sequences of $P + N - 1$ half-planes that satisfy all conditions of Proposition 10.

4. For each found cycle, choose all possible signs of aa-saddles: get all possible embeddable essential discs with (P, N) vertices up to easy conjugations.

The first part of this algorithm has been implemented using the algorithm of [27]: it corresponds to listing all *necklaces* of *length* $l = P + N$, with number of *colors* $k = 2$ (positive and negative vertices), and *density* $d = P$ (the number of non-zero colors). We have implemented the algorithm as a GAP procedure, called EnumerateNecklaces(l, k, d), which gives us all vertex strings V with (P, N) vertices up to cycling.

Then, the procedure EnumerateHalfPlanes(V) lists all embeddable essential regular half-planes in terms of their b-arcs.

To list all good H_θ-sequences for a given vertex string V, we consider the directed graph G with nodes all the regular essential embeddable half-planes with vertex string V, and (directed) edges* the saddles occurring among them: notice that aa-saddles are loops: edges going from one node to itself. This is made by our GAP procedure MakeGraph(V).

Now, a good H_θ-sequence corresponds to a cycle in G, of length $P+N-1$, such that the above conditions are satisfied. Such a cycle never passes twice through the same edge, because it would pass twice through the same saddle with the same sign. This observation is the key to our enumeration algorithm for H_θ-sequences, which is invoked by the GAP function EnumerateCycles(G, n), which enumerates the first n H_θ-sequences on the graph G, or all of them if $n = 0$ (see the Appendix for the details of the algorithm).

The whole process is performed in one step by ComputeCycles(P, N), which computes all different cycles starting from a given number of positive and negative vertices.

Question 6. We noticed that many vertex strings have no associated H_θ-sequence. This may happen for various reasons: There may be too few half-planes and saddles, or the resulting graph might be too disconnected, or have too few different saddles. We suggest that this matter be investigated, with the goal of discarding some vertex strings a priori, perhaps reducing considerably the running time of the algorithm. For example, vertex strings in which positive and negative vertices alternate too closely have very few or no essential half-planes associated to them. To give an

*We refrain from using the more standard term "arc" for obvious reasons.

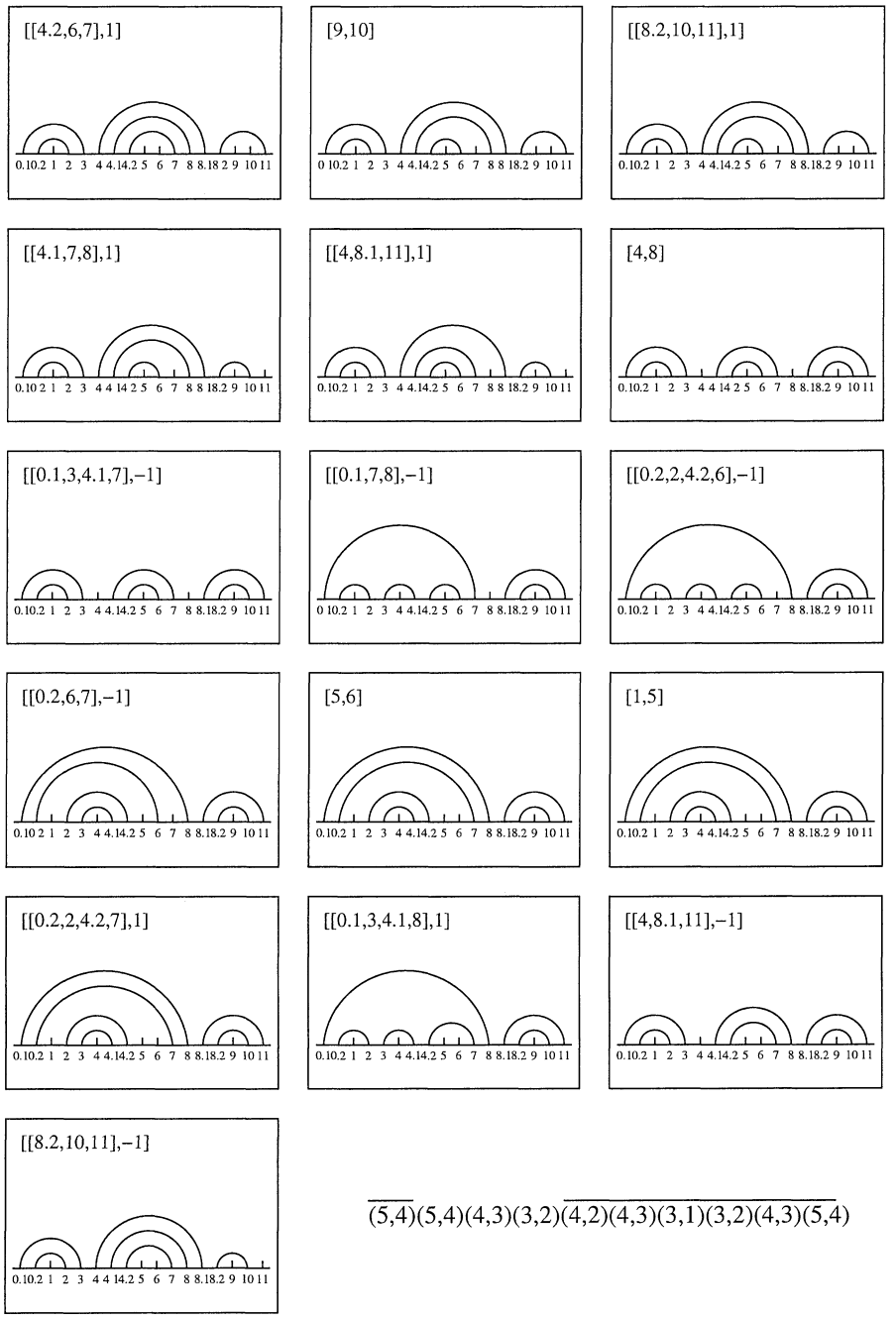

Fig. 22. The H_θ-sequence of the disc of Figure 9.

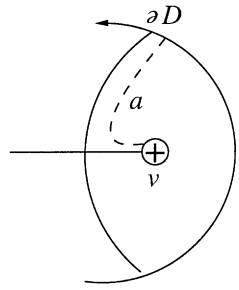

Fig. 23. A vertex of type (a).

example: There are 43 vertex strings with $(8,4)$ vertices, but only 14 of them have some cycle associated.

Remark: An **end-tile** on an embeddable foliated disc is a part of the disc which contains a positive vertex which is only attached to one aa-saddle. For example, our disc of Figure 6 has two end-tiles: saddle 11 and saddle 2. These saddles can easily be eliminated by a Markov move, also reducing the braid index. This can be easily seen on the extended word: if one index only occurs in one letter of EW, it is surely corresponding to one end-tile; we can remove that letter and dropping all following indices by 1, getting another word which is the extended word of a simpler essential embeddable disc. Our ComputeCycles(P,N) tells us which of these H_θ-sequences are associated to discs without end-tiles.

How does the removal of an end-tile change the corresponding boundary word? If it was $EW \in B_P, BW \in B_n, n = P - N$, taking away the end-tile and the corresponding strand, we get a new $EW' \in B_{P-1}, BW' \in B_{n-1}, n-1 = P-1-N$. The subword of BW corresponding to the letter of EW we have deleted does not occur any more in BW'; the indices which have been dropped by 1 in EW' must be dropped by 1 in BW' as well. For instance, in our example of Figure 13, if we cancel from EW the second letter we get the new boundary word

$$BW' = \delta_{5,2}(5,2)\delta_{4,2}(5,4)(4,1)\delta_{5,2}^{-1}\delta_{5,2}^{-1}.$$

How to draw the H_θ-sequence: We have written a GAP procedure PSFilm(C,G), which, given a cycle C for the graph G, outputs a PostScript file with the drawing of the $P+N-1$ half-planes, each with its vertices and b-arcs, the saddle occurring between two consecutive of them (with no sign for the aa-saddles) and a boundary word resulting from arbitrarily assigning signs to aa-saddles. For instance, the H_θ-sequence of our disc of Figure 9 can be seen in Figure 22.

There is also the possibility of exporting in GML (Graph Modelling Language) a drawing of the tiled disc corresponding to a cycle C *via* the procedure DrawDisc(C,G).

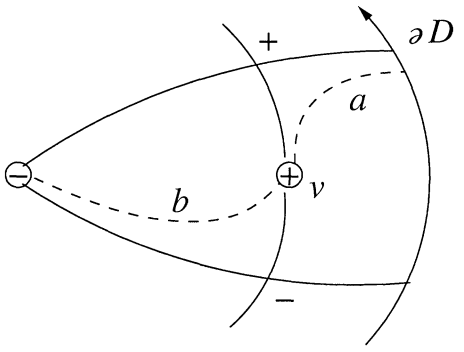

Fig. 24. A vertex of type (a, b) and sign $(+, -)$.

The resulting file is readable by GML-aware software, such as Graphlet [9]. Usually standard planar-graph layout algorithms are able to display such tiled discs correctly, but sometimes a bit of tweaking is required. We plan to implement in the future a more sophisticated layout algorithm that uses the known cycling order of the vertices.

7. Some interesting data

The main theorem in [6] is the basis for the algorithm in [2]. This theorem was later re-proved as Theorem 4.3 of [1], which we now present. Before so-doing we need several definitions.

To each tiled disc we can associate the *graph of singular leaves*, in which we consider as vertices of the graph only the (P, N) intersection points with the axis.

The *valence* of a vertex in the graph is the number of singular leaves which meet at that vertex. Each non-singular leaf which has an endpoint at the vertex is necessarily type a or type b, with the type of that leaf changing only after the passage through a singular leaf. We define the *type* of the vertex to be the cyclic array of a's and b's which describes the non-singular leaf types as we travel around the vertex in the order in which they are encountered in the fibration. The *sign* of a vertex (as vertex of the graph) is the cyclic array of signs of the singular leaves as we travel around the vertex, again ordered by the order in which they are encountered in the fibration. The main theorem in [6], which is also Theorem 4.3 of [1], asserts:

Theorem 7. Let D be an embeddable disc which supports a braid foliation. Then there is a sequence of embeddable foliated discs:

$$D = D_1 \to D_2 \to \cdots D_k$$

such that D_k has a radial foliation, without singularities, and the graph of singular leaves for D_{i+1} is obtained from that for D_i by one of the following:

1. The graph of singular leaves for D_i contains a vertex v of valence 1 (see Figure 23). Delete v and the unique singular leaf which ends at v.

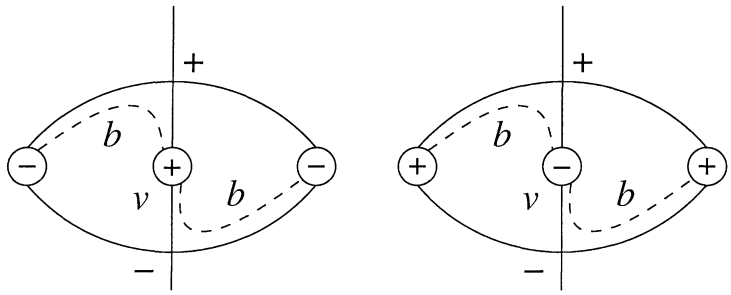

Fig. 25. Two vertices of type (b, b) and sign $(+, -)$.

2. The graph of singular leaves for D_i contains a vertex of valence 2, type (a, b) and sign $(+, -)$ (see Figure 24). Do an ab-exchange move, as defined in [1]. This move deletes two vertices of opposite sign and two singularities of opposite sign from the foliation.

3. The graph of singular leaves for D_i contains a vertex of valence 2, type (b, b) and sign $(+, -)$ (see Figure 25). Do a bb-exchange move as defined in [1]. This move deletes two vertices of opposite sign and two singularities of opposite sign from the foliation.

The theorem leads to a natural question: Is this set of moves "minimal", or can we eliminate one or more of them? The data collected in this paper helps to begin to answer that question.

Question 7. Do there exist examples of embeddable foliated discs which have no vertices of valence 1, i.e. which have no end-tiles? If not, then Theorem 7 could obviously be simplified by eliminating moves 2 and 3.

At the time when Theorem 7 was proved, we knew, a classical theorem due to Magnus and Pelluso [21] that there are no examples with $n \leq 3$. We also knew, form Morton's work in [24], that such examples *exist* when the braid index is 4. Discs whose boundaries have braid index 4 will have $N \geq 0$ negative vertices and $4 + N$ positive vertices, so we searched. There are no such examples of 4-braids with $(P, N) = (5, 1), (6, 2)$ or $(7, 3)$. We found 16 examples with $(P, N) = (8, 4)$, and (to our great surprise) none with $(P, N) = (9, 5)$ or $(10, 6)$. The data suggest that there is structure, not yet understood.

Question 8. Do there exist examples of embeddable foliated discs which have no vertices of valence 1 and no vertices of valence 2, type (b, b), sign $(+, -)$?

We do not know the answer to this question. All of our 4-braid examples which lack vertices of valence 1 have vertices of valence 2 and both type (a, b) and type (b, b) with sign $(+, -)$.

Question 9. Do there exist examples of embeddable foliated discs which have no vertices of valence 1 and no vertices of valence 2, type (a, b), sign $(+, -)$? Our final

example says "yes" (see Figure 28). To find it we had to go to $(P, N) = (13, 4)$. This example tells us that we cannot eliminate move 3.

We do not know whether simpler examples of the same type exist. There are hints of much more structure in the first set of examples, but we don't have enough data to say more at this time.

Question 10. Can move 3 be eliminated in the special case $n = 4$, i.e. $(P, N) = (N + 4, N)$? (Notice that the counterexample in Figure 28 has braid index 9).

The data There is only one vertex string for $(P, N) = (5, 1)$. Our GAP procedure ComputeCycles(5,1) gives for this 64 cycles, none of which is without end-tiles.

There are four vertex strings for $(P, N) = (6, 2)$. Our GAP procedure ComputeCycles(6,2) gives for them 276 cycles, none of which is without end-tiles.

There are 12 vertex strings with $(7, 3)$ vertices, but 4 of them have no associated cycles. For the other 8 strings, our GAP procedure ComputeCycles(7,3) gives 828 cycles, none of which is without end-tiles.

The first interesting discs for B_4 are those with $(8, 4)$ vertices: there are 43 vertex strings, 29 of which have no associated cycles; for the other 14 vertex strings, our GAP procedure ComputeCycles(8,4) gives 2944 cycles, 16 of which are with no end-tiles: all these 16 have the same vertex string (which also has other 12 discs with end-tiles). The 16 discs without end-tiles are the following 8 and their inverses:

1. $D_1 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [3.1, 6, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], \overline{[1, 2, 5.1]}, \overline{[2, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
2. $D_2 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [3.1, 6, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], \overline{[1, 3, 5.1]}, [1, 2], [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
3. $D_3 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [3.1, 6, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], [2, 3], \overline{[1, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
4. $D_4 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [3.1, 6, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [4, 5], [2, 3.1, 5], \overline{[1, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
5. $D_5 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [3.1, 7, 8], [3.1, 6, 7], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], \overline{[1, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
6. $D_6 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [3.1, 6, 8], [7, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], \overline{[1, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
7. $D_7 = \{\overline{[3.1, 5, 8]}, \overline{[4, 5.1, 7]}, [6, 7], [3.1, 6, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], \overline{[1, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$,
8. $D_8 = \{\overline{[3.1, 5, 8]}, [4, 5], \overline{[4, 5.1, 7]}, [3.1, 6, 8], \overline{[0.1, 3, 7]}, \overline{[0.2, 2, 3.1, 6]}, [1, 4, 5.1], [2, 3.1, 5], \overline{[1, 3, 5.1]}, [0.2, 2, 6], [0.1, 3, 5.1, 7]\}$.

Their corresponding boundary words, depending on the sign assigned to the aa-saddle, are the following:

1. $BW_1 = (4,3)\overline{(3,2)(4,3)(4,3)}(3,2)\overline{(2,1)}(3,2)(2,1)\overline{(3,2)}$,
2. $BW_2^- = BW_1, (EW_2^- = r_8(EW_1))$,
3. $BW_2^+ = (4,3)\overline{(3,2)(4,3)(4,3)}(3,2)\overline{(2,1)}(3,2)(2,1)(2,1)\overline{(3,2)}$,
4. $BW_3^- = BW_1, (EW_3^- = r_8(EW_1))$,
5. $BW_3^+ = BW_2^+, (EW_3^+ = r_8(EW_2^+))$,
6. $BW_4^- = (4,3)\overline{(3,2)(4,3)(4,3)}(3,2)(2,1)\overline{(3,2)}$,
7. $BW_4^+ = (4,3)\overline{(3,2)(4,3)(4,3)}(3,2)\overline{(2,1)}(3,2)(3,2)(2,1)(2,1)\overline{(3,2)}$,
8. $BW_5 = (4,3)\overline{(3,2)(4,3)}(3,2)\overline{(2,1)}(3,2)(2,1)(2,1)\overline{(3,2)}$,
9. $BW_6^- = (4,3)\overline{(3,2)(4,3)(4,3)(4,3)}(3,2)\overline{(2,1)}(3,2)(2,1)(2,1)\overline{(3,2)}$,
10. $BW_6^+ = BW_5, (EW_6^+ = r_3(EW_5))$,
11. $BW_7^- = BW_6^-, (EW_7^- = r_3(EW_6^-))$,
12. $BW_7^+ = BW_5, (EW_7^+ = r_3(EW_5))$,
13. $BW_8^- = (4,3)\overline{(3,2)(3,2)(4,3)(4,3)}(3,2)\overline{(2,1)}(3,2)(2,1)(2,1)\overline{(3,2)}$,
14. $BW_8^+ = \overline{(4,3)}(3,2)\overline{(2,1)}(3,2)(2,1)(2,1)\overline{(3,2)}$.

The conjugacy classes have been computed by S. J. Lee: considering the 16 words (the 8 which are different in this list, and their inverses), almost all of them are non conjugate. There are only two pairs of conjugate braids: $BW_4^- \sim (BW_8^+)^{-1}$ and their inverses.

The disc $(D_7^-)^{-1}$, that can be seen in Figure 26, is the disc corresponding to Morton's braid, as shown by G. Wright in [29].

All other discs have a similar structure, with two 'squares' of vertices joined by an aa-saddle; only discs D_1, D_5 and their inverses have a different structure: a pentagon joined directly to a square, as can be seen in Figure 27.

There are 9,288 cycles with $(9,5)$ vertices, but none of them is without end-tiles. There are 37,952 cycles with $(10,6)$ vertices, but none of them is without end-tiles.

An interesting disc with $(13,4)$ vertices, found by Birman and Menasco, is the following (see Figure 28):

$$D = [[[0.1,4,5.1,8],-1],[[3,0.2,7,5.2],-1],[2,6],[1,6],[[3,0.2,7,5.2],1],$$
$$[[0.1,4,5.1,8],1],[5,13],[[0.1,4,12],-1],[[3,0.2,11],-1],[5,10],[1,9],$$
$$[2,10],[[3,0.2,11],1],[[0.1,4,12],1],[12,13],[9,11]].$$

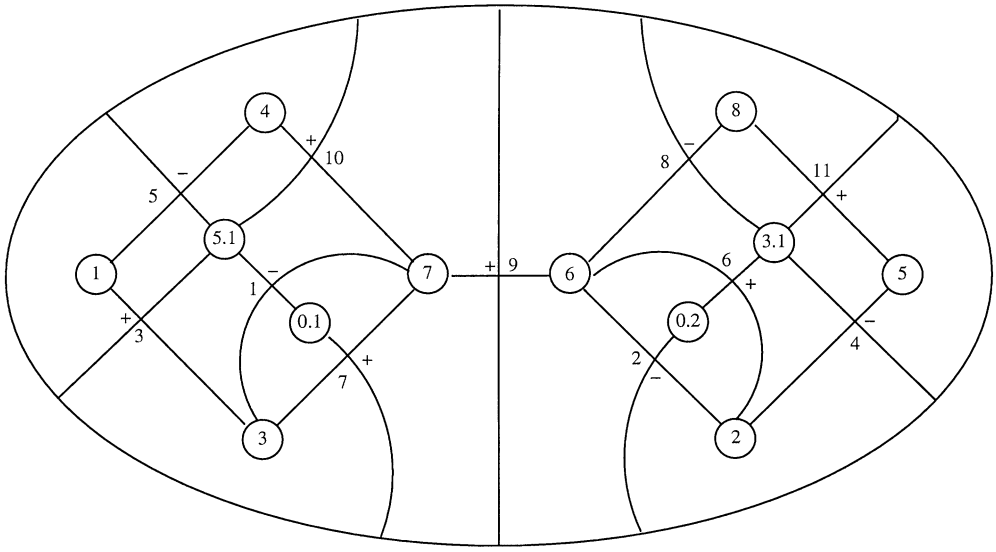

Fig. 26. The disc $(D_7^-)^{-1}$.

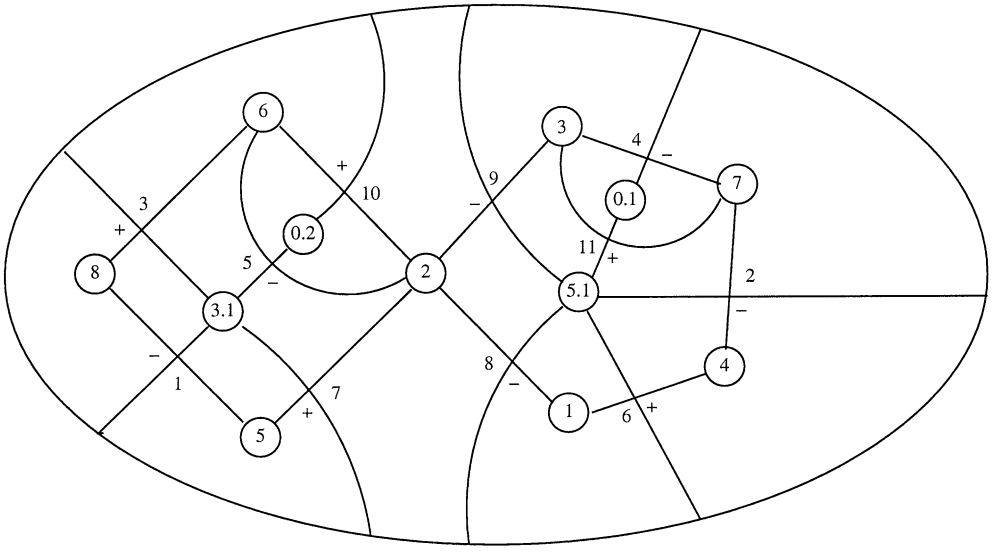

Fig. 27. The disc D_1.

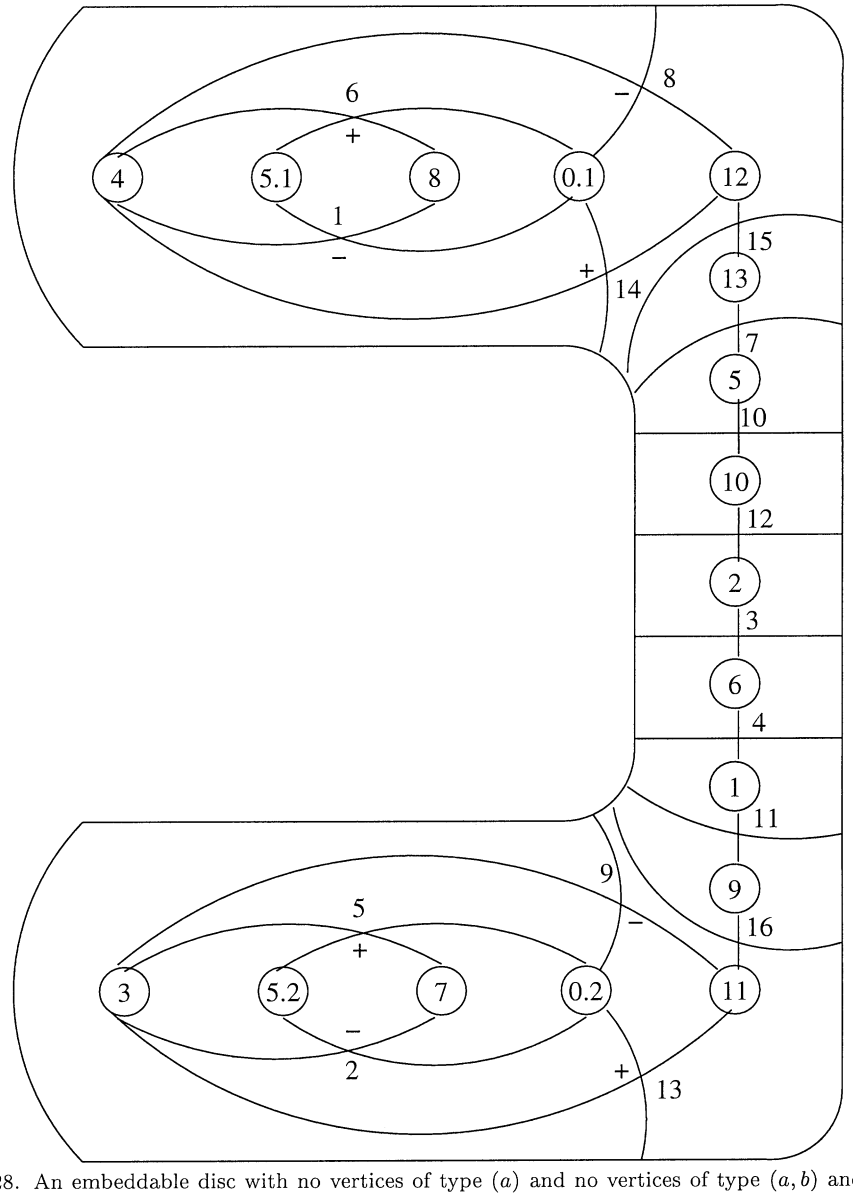

Fig. 28. An embeddable disc with no vertices of type (a) and no vertices of type (a, b) and sign $(+, -)$.

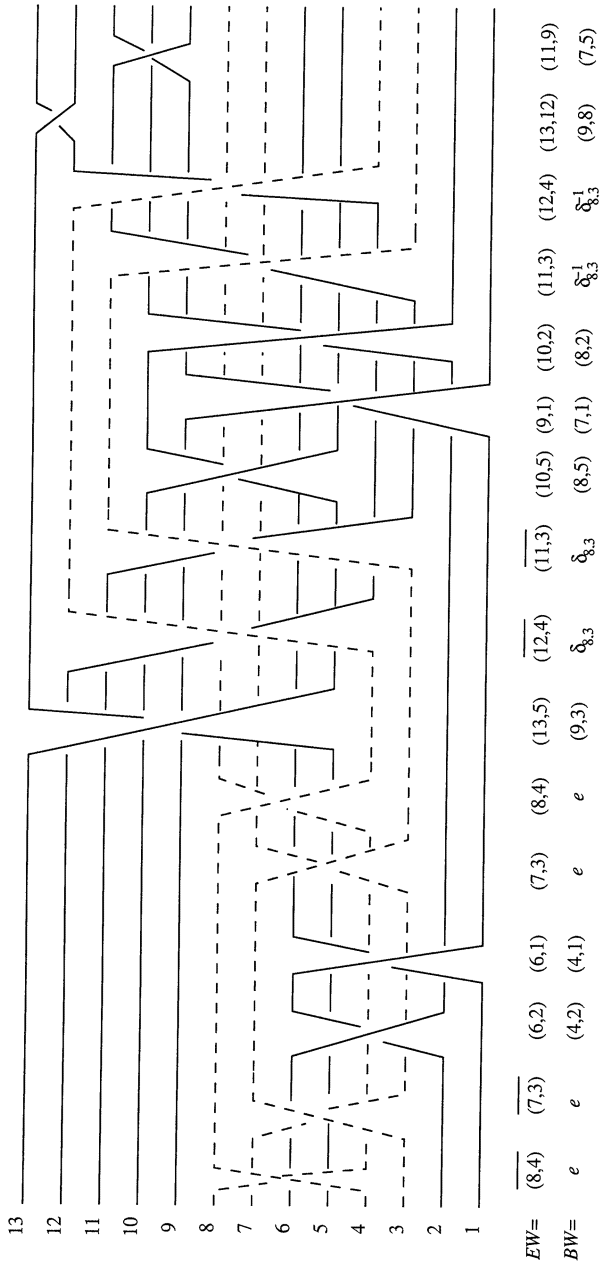

Fig. 29. The extended braid EW; the dashed strands have to be removed to get the boundary braid BW.

$P = 13, N = 4, n = 9, P + N - 1 = 16$, number of saddles.

$$EW = \overline{(8,4)(7,3)}(6,2)(6,1)(7,3)(8,4)(13,5)\overline{(12,4)(11,3)}(10,5)(9,1)$$

$$(10,2)(11,3)(12,4)(13,12)(11,9).$$

$$\rho(W) = (1,6,10,5,12,13,2,11,9)(3)(4)(7)(8).$$

Deleting the four strands corresponding to the four 1-cycles of $\rho(W)$, as can be seen in Figure 29, get the 9-braid

$$BW = (4,2)(4,1)(9,3)\delta_{8,3}\delta_{8,3}(8,5)(7,1)(8,2)\delta_{8,3}^{-1}\delta_{8,3}^{-1}(9,8)(7,5) =$$

$$= (4,2)(4,1)(9,3)(8,7)(7,6)(6,5)(5,4)(4,3)(8,7)(7,6)(6,5)(5,4)(4,3)(8,5)$$

$$(7,1)(8,2)\overline{(4,3)(5,4)(6,5)(7,6)(8,7)(4,3)(5,4)(6,5)(7,6)(8,7)}(9,8)(7,5).$$

Reference

[1] J. S. Birman, E. Finklestein, Studying surfaces via closed braids, *Journal of Knot Theory and its Ramifications*, vol. 7, n. 3 (1998) 267–334.

[2] J. S. Birman, M. D. Hirsch, A new algorithm for recognizing the unknot, *Geometry and Topology*, 2 (1998) n. 9, 175–220.

[3] J. S. Birman, K. H. Ko, S. J. Lee, A new approach to the word and conjugacy problems in the braid groups, *Advances in Mathematics*, 139 (1998), pp. 322–353.

[4] J. S. Birman, K. H. Ko, S. J. Lee, "The infimum, supremum and geodesic length of a braid conjugacy class", preprint arXiv:math.GT/0003125

[5] J. S. Birman, W. Menasco, Studying links via closed braids IV: composite links and split links, *Inventiones math.* 102 (1990), 115-139.

[6] J. S. Birman, W. Menasco, Studying links via closed braids V: the unlink, *Transactions A.M.S.* 329 (1992) 585–606.

[7] T. Fiedler, A small state sum for knots, *Topology* 32, No. 2 (1993), 281-294.

[8] H. Goeritz, Bemerkunger zur Knotentheorie, *Hamburg Abhandlungen* 10 (1934), 201-210.

[9] Graphlet: A toolkit for graph editors and graph algorithms, Faculty of Mathematics and Computer Science, University of Passau, `http://www.infosun.fmi.uni-passau.de/Graphlet/`.

[10] W. Haken, Theorie der Normalflachen, *Acta Math.* 105 (1961) 245–375.

[11] J. Hass, Algorithms for Recognizing Knots and 3-Manifolds, *Chaos Solitons Fractals* 9 (1998) 569–581.

[12] J. Hass and J. Lagarias, The number of Reidemeister moves needed for unknotting, *J. Amer. Math Soc.* 14 (2001) 399–428.

[13] J. Hass, J. Lagarias and N. Pippenger, The computational complexity of knot and link problems, *J. ACM* 46 (1999) 185–211.

[14] J. Hoste, M. Thistlethwaite and J. Weeks, The first 1,701,936 knots, *The Mathematical Intelligencer*, 20, No. 4 (1998), 33–48.

[15] W. Jaco and J.L.Tollefson, Algorithms for the complete decomposition of a closed 3-manifold, *Ill J. Math* 39 (1995), 358-406.

[16] D. B. Johnson, Finding all the elementary circuits of a directed graph, *SIAM J. Comput.* 4 (1975) 77–84.

[17] E.S.Kang, K. H. Ko, S. J. Lee, Band presentation for the 4-braid group, *Topology and its Applications* 78 (1997) 39-60.

[18] H. Kneser, Geschlossene Flachen in dreidimensionalen Mannigfaligkeiten, *Jahresbericht Math. Verein.* 28 (1929), 248-260.
[19] K. H. Ko, S. J. Lee, Genera of some closed 4-braids, *Topology and its Applications* 78 (1997) 61–77.
[20] S.J. Lee, Implementation of an algorithm for solving the conjugacy problem in the braid groups.
[21] W. Magnus and A. Pelluso, On a theorem of V.H.Arnold, *Com. on Pure and Applied Math.*, XXII (1969), 683-692.
[22] H. R. Morton, Exchangeable Braids, *Low–dimensional Topology*, London Math. Soc., 1982, Lecture Notes in Science vol 95, ed. R. Fenn, pp. 86–105.
[23] H. R. Morton, Threading knot diagrams, *Math. Proc. Cambridge Phylos. Soc.* 93 (1983) 247–261.
[24] H. R. Morton, An irreducible 4–braid with unknotted closure, *Math. Proc. Camb. Phil. Soc.* 93 (1993) 259–261.
[25] M. Rampichini, *Exchangeable fibred links*, doctoral thesis, february 1998, Milano, Italy.
[26] L. Rudolph, Quasipositive plumbing (Constructions of quasipositive knots and links, V), *Proc. AMS* 126 (1998) 257–267.
[27] F. Ruskey, J. Sawada, An Efficient Algorithm for Generating Necklaces with Fixed Density, *SIAM J. Comput.*, 29 (1999) 671–684.
[28] P. Vogel, Representation of links by braids: a new algorithm, *Comm. Math. Helv.*, 65 (1990) 104–113.
[29] G. Wright, *A foliated disk whose boundary is Morton's irreducible 4-braid*, preprint, March 1998.
[30] S. Yamada, The minimal number of Seifert circles equals the braid index of a link, *Invent. Math.* 89 (1987), 347–356.

Appendix A. Commented List of Main GAP Procedures and Functions.

`EnumeratePositiveGoodWords(p)`

Enumerates the positive good words (braids) for `p` vertices up to conjugations by δ and by subwords. The number of such words is inherently exponential in `p`, so even if the test for easy conjugations, that is, the test for equivalence modulo permutation of indices and of letters is easy (quadratic in `p`), the overall complexity of the procedure is not polynomial in `p`.

`EnumerateGoodWords(p)`

Enumerates the good words (braids) by suitably inserting signs in the words obtained through `EnumeratePositiveGoodWords(p)`. Each word is a list of pairs; each pair is formed by the generator name (a pair of increasing indices) and a sign.

`GenerateDiscBoundary(n, w)`

Takes a good word `w` with `n` indices as input and gives back the boundary-point list of the corresponding tiled disc. It also checks whether the word is really a good one, and returns fail if this is not true.

`GetInsertionArcs(n, b)`

Takes the braid index and the list of boundary points of an extended word and gives back a list of insertion intervals as records with fields `i`, `f`, and `I`, where `i` is the initial point of the insertion arc, `f` is the final point (they may coincide) and `I` is the related list of intervals of possible positions for a new negative vertex.

A disc boundary point is described as a record with fields `s` and `l`, where `s` describes the saddle associated with the boundary point and `l` is its level; the saddle is described by a pair whose first coordinate is the list of vertices involved in the saddle (starting from a positive vertex, and with, if possible, alternating signs), and the second one is the sign.

`GetSaddles(n, s, b)`

Takes the braid index `n` of an extended word, the number `s` of saddles of the disc generating the extended word and the list `b` of boundary points (in the same format of `GetInsertionArcs()`, and gives back a list of records with the following fields: `i` and `f` are the initial and final positive vertices of the insertion arc; `N` is a list of cyclic intervals, indicating the possible position of the new negative vertex; `S` is a list of cyclic intervals, indicating the possible levels of the new saddle; the sign `s` is the possible sign of the new saddle, `I` is a subinterval of `N` where the saddle can exist with sign `s`, `pi` and `pf` are initial and final points (given by their order on the boundary) of the insertion arc.

`Embeddable(n, nsaddles, boundary, bbsaddles)`

Takes the number `n` of positive vertices of a disc, the number `nsaddles` of saddles, the list `boundary` of boundary points and the list `bbsaddles` of *bb*-saddles, and outputs whether or not this tiled disc is embeddable. `bbsaddles` has the same format as `boundary`, but the order is not relevant.

`InsertVertices(n, nsaddles, boundary, bbsaddles, depth)`

Takes the number `n` of positive vertices of a disc, the number `nsaddles` of saddles, the list `boundary` of boundary points, list `bbsaddles` of *bb*-saddles, a limit `depth` on the number of negative vertices to add (if equal to -1, it tries all possible insertions) and returns a list of embeddable discs with additional negative vertices; each disc is a record with fields `b`, containing the list of boundary points, and `bb`, containing the list of *bb*-saddles.

`EnumerateNecklaces(n, k, d)`

Returns the set of necklaces of length `n`, `k` colours and density (number of nonzeros) `d` using the Ruskey–Sawada algorithm [27].

Even if the number of necklaces is superpolynomial in their length, the Ruskey–Sawada algorithm has constant amortized complexity, that is, the time required

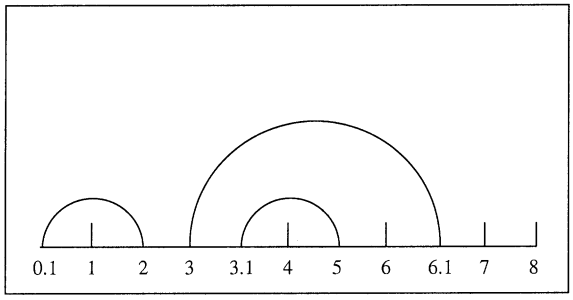

Fig. A.1. An embeddable essential halfplane.

to generate a single necklace is constant, which implies that the time necessary to enumerate all necklaces is linear in their number.

`EnumerateHalfPlanes(vertexString)`

Generates all possible embeddable essential halfplanes on the given string of vertices, which must be a vector of zeroes (negative vertices) and ones (positive vertices). Returns a record with two fields, `halfPlane`, described below, and `vertexString`, which copies the input field.

The field `halfPlane` is a list of vectors indexed by the vertices. The i-th component of a vector is a record with fields `v` and `c`, where `v` is the vertex linked by an arc to vertex i and `c` the index of the "outer" connected component `v` belongs to (the vertices are numbered all from 1 to $P + N$; 0 means no connection with any vertex). For instance `[ rec( v := 3, c := 0 ), rec( v := 0, c := 1 ), rec(v := 1, c := 0 ), rec( v := 9, c := 0 ), rec( v := 7, c:= 2 ), rec( v := 0, c := 3 ), rec( v := 5, c := 2 ), rec( v := 0, c := 2 ), rec( v := 4, c := 0 ),rec( v := 0,c := 0 ),rec( v := 0, c := 0 ) ]` describes the half-plane of Figure A.1.

`MakeGraph(halfPlaneRecord)`

Builds the halfplane adjacency graph for the specified set of halfplanes. The result is a record with fields `halfPlane`, `vertexString`, `graph`, `N`, `A` and `saddle`. `graph` is an adjacency list: to each halfplane i we associate a list of records with fields `v`, `s` and `a` where `v` is the index of the next halfplane, `s` is the associated saddle index (in the list `saddle`) multiplied by its sign, and `a` is the index, in the `v`-th list, of the symmetric edge.

The list `saddle` is a list of vectors of variable length (2, 3 or 4) indicating which vertices are involved in a saddle. The vertices are given by their indices in `halfPlaneRecord.vertexString`.

The fields `N` and `A` hold the number of nodes and arcs of the generated graph.

The graph is generated by applying a simple test to each pair of nodes (halfplanes); thus, the generation requires time quadratic in the number of halfplanes.

`MakeRandomGraph(halfPlaneRecord, isRandom)`

Same as `MakeGraph()`, but `isRandom` is a boolean indicating whether we want to scramble randomly the halfplane order. This is useful for single-cycle generation.

If `isRandom` is false the halfplanes are used in the order in which they appear in `halfPlaneRecord`. Otherwise, they are permuted randomly. The GAP pseudorandomness seed values `R_N` and `R_X` are accumulated into the two additional fields of the resulting record with the same name.

`EnumerateCycles(halfPlaneGraph, stopAt)`

This function is the core of the enumeration process; it enumerates the H_θ-sequences related to the provided `halfPlaneGraph`; the sequences are expressed as lists of records with fields `v` and `s`, where `v` is a node and `s` is the signed index of the saddle that labels the edge towards the next node of the cycle (however, aa-saddles have always positive sign).

The parameter `stopAt` specifies how many cycles to generate. If it is zero, all cycles will be generated. Otherwise, only `stopAt` cycles will be generated. By applying this function with `stopAt=1` on randomly scrambled graphs, it is possible to generate longer cycles than those allowed by exhaustive enumeration.

The algorithm used in this function is based on the notion of **dual graph**: given a directed graph G with node set V and directed-edge set A, the dual graph has as node set A and directed-edge set $B \subseteq A \times A$, where $(a, a') \in B$ iff the target of a is the source of a'. More precisely, if we call $s : A \to V$ and $t : A \to V$ the (obvious) **source** and **target** functions of G, we can build the pullback

$$\begin{array}{ccc} B & \longrightarrow & A \\ \downarrow & & \downarrow s \\ A & \xrightarrow{t} & V \end{array}$$

Then, the dual graph has directed-edge set B, and the projections of the pullback are precisely its source and target functions. The crucial observation is that *elementary* cycles in the dual graph (i.e., cycles that never pass twice through the same *node*) are in bijection with cycles of the original graph that never pass twice through the same *edge*. This allows one to use standard enumeration methods for elementary cycles for enumerating H_θ-sequences.

Nevertheless, sophisticated methods such as Johnson's algorithm [16] turn out to be inefficient in our case: this happens because the length of the cycles to be generated is fixed, and is very small with respect to the size of the graph. After several experiments, we focused on a two-phase visiting algorithm. In the first phase, we use standard depth-first enumeration techniques to generate candidate loop-free cycles of length shorter than or equal to $P + N - 1$; in the second phase, we enrich each candidate with loops, and test the various conditions that must be satisfied to obtain an H_θ-sequence.

In the first phase, a total ordering is established on the nodes of the dual graph. At each round of the generation, a *root* node of the dual graph is chosen as first node of the cycle to be generated; moreover, the generation process only considers nodes larger than the root. This ensures that loop-free cycles are never generated twice. Moreover, a precomputation of the distances from each node to the root (using a standard breadth-first visit) allows to cut prematurely cycles that could never "get back in time" because they have moved too far apart from the root.

In the second phase (which is invoked for each candidate loop-free cycle) we exhaustively try to add loops so to obtain an H_θ-sequence.

Note that the number of cycles to be generated is inherently superexponential; it is also very difficult to estimate the amortized complexity of the enumeration process. The division in two phases cuts a large part of the search space with respect to a simple enumeration, but nonetheless exhaustive enumeration is possible only for a very small number of vertices.

`ComputeCycles(p, n)`

This function generates all cycles for `p` positive and `n` negative vertices, using the functions above. The result is provided in two fields named `c` and `e`, which contains all cycles, and all cycles that do not contain end tiles, respectively. A cycle here is specified in a more user-friendly form, that is, as a sequence of saddles, each saddle being of the form `[v,w]` for aa-saddles, and of the form `[[v,w,z],s]` or `[[u,v,w,z],s]` for ab and bb-saddles (`s` is the sign of the saddle). Vertices are named as in this paper.

`PSFilm(c, halfPlaneGraph)`

Outputs in the current directory a file named `film.ps` containing a PostScript visualization of the sequence of halfplanes traversed by the H_θ-sequences represented by the cycle `c`. The visualization also contains one of the corresponding boundary braid words (aa-saddles signs are arbitrary).

`DrawDisc(c, halfPlaneGraph)`

Outputs in the current directory a file named `disc.gml` containing the graph structure of the tiled disc associated to the cycle `c`. Note that the graph is *not yet* embedded in the plane: a planar embedding layout algorithm must be applied.