



HAL
open science

Local Certification of Graphs with Bounded Genus

Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Eric Rémila, Ioan Todinca

► **To cite this version:**

Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Eric Rémila, et al.. Local Certification of Graphs with Bounded Genus. *Discrete Applied Mathematics*, 2023, 325, pp.9–36. 10.1016/j.dam.2022.10.004 . hal-03663680

HAL Id: hal-03663680

<https://hal.science/hal-03663680v1>

Submitted on 10 May 2022



HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Local Certification of Graphs with Bounded Genus

Laurent Feuilloley  

Univ. Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France

Pierre Fraigniaud  

IRIF, CNRS and Université de Paris, France

Pedro Montealegre  

Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile,

Ivan Rapaport  

DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Chile

Éric Rémila  

Univ Lyon, UJM Saint-Etienne, GATE L-SE UMR 5824, F-42023 Saint-Etienne, France

Ioan Todinca  

LIFO, Université d'Orléans and INSA Centre-Val de Loire, France

Abstract

Naor, Parter, and Yogev [SODA 2020] recently designed a compiler for automatically translating standard centralized interactive protocols to *distributed* interactive protocols, as introduced by Kol, Oshman, and Saxena [PODC 2018]. In particular, by using this compiler, every linear-time algorithm for deciding the membership to some fixed graph class can be translated into a $\text{dMAM}(O(\log n))$ protocol for this class, that is, a distributed interactive protocol with $O(\log n)$ -bit proof size in n -node graphs, and three interactions between the (centralized) computationally-unbounded but non-trustable *prover* Merlin, and the (decentralized) randomized computationally-limited *verifier* Arthur. As a corollary, there is a $\text{dMAM}(O(\log n))$ protocol for recognizing the class of planar graphs, as well as for recognizing the class of graphs with bounded genus.

We show that there exists a distributed interactive protocol for recognizing the class of graphs with bounded genus performing just a *single* interaction, from the prover to the verifier, yet preserving proof size of $O(\log n)$ bits. This result also holds for the class of graphs with bounded *non-orientable genus*, that is, graphs that can be embedded on a *non-orientable* surface of bounded genus. The interactive protocols described in this paper are actually *proof-labeling schemes*, i.e., a subclass of interactive protocols, previously introduced by Korman, Kutten, and Peleg [PODC 2005]. In particular, these schemes do *not* require any randomization from the verifier, and the proofs may often be computed a priori, at low cost, by the nodes themselves. Our results thus extend the recent proof-labeling scheme for planar graphs by Feuilloley et al. [PODC 2020], to graphs of bounded genus, and to graphs of bounded non-orientable genus.

2012 ACM Subject Classification D.1.3 Concurrent Programming (Distributed programming); F.2.2 Nonnumerical Algorithms and Problems.

Keywords and phrases Local certification, proof-labeling scheme, locally checkable proofs

Funding *Laurent Feuilloley*: MIPP and ANR project GrR

Pierre Fraigniaud: ANR project DESCARTES, and INRIA project GANG

Pedro Montealegre: ANID via PAI + Convocatoria Nacional Subvención a la Incorporación en la Academia Año 2017 + PAI77170068 and FONDECYT 11190482

Ivan Rapaport: CONICYT via PIA/Apoyo a Centros Científicos y Tecnológicos de Excelencia AFB 170001 and Fondecyt 1170021

Éric Rémila: IDEX LYON (project INDEPTH) within ANR-16-IDEX-0005 and MODMAD

1 Introduction

1.1 Context and Objective

The paper considers the standard setting of distributed network computing, in which processing elements are nodes of a network modeled as a simple connected graph $G = (V, E)$, and the nodes exchange information along the links of that network (see, e.g., [45]). As for centralized computing, distributed algorithms often assume promises on their inputs, and many algorithms are designed for specific families of graphs, including regular graphs, planar graphs, graphs with bounded arboricity, bipartite graphs, graphs with bounded treewidth, etc. Distributed *decision* refers to the problem of checking that the actual input graph (i.e., the network itself) satisfies a given predicate. One major objective of the check up is avoiding erroneous behaviors such as livelocks or deadlocks resulting from running an algorithm dedicated to a specific graph family on a graph that does not belong to this family. The decision rule typically specifies that, if the predicate is satisfied, then all nodes must accept, and otherwise at least one node must reject. A single rejecting node can indeed trigger an alarm (in, e.g., hardwired networks), or launch a recovery procedure (in, e.g., virtual networks such as overlay networks). The main goal of distributed decision is to design efficient checking protocols, that is, protocols where every node exchange information with nodes in its vicinity only, and where the nodes exchange a small volume of information between neighbors.

Proof-Labeling Schemes.

Some graph predicate are trivial to check locally (e.g., regular graphs), but others do not admit local decision algorithms. For instance, deciding whether the network is bipartite may require long-distance communication for detecting the presence of an odd cycle. *Proof-labeling schemes* [34] provide a remedy to this issue. These mechanisms have a flavor of NP-computation, but in the distributed setting. That is, a non-trustable oracle provides each node with a *certificate*, and the collection of certificates is supposed to be a *distributed proof* that the graph satisfies the given predicate. The nodes check locally the correctness of the proof. The specification of a proof-labeling scheme for a given predicate is that, if the predicate is satisfied, then there must exist a certificate assignment leading all nodes to accept, and, otherwise, for every certificate assignment, at least one node rejects. As an example, for the case of the bipartiteness predicate, if the graph is bipartite, then an oracle can color blue the nodes of one of the partition, and color red the nodes of the other partition. It is then sufficient for each node to locally check that all its neighbors have the same color, different from its own color, and to accept or reject accordingly. Indeed, if the graph is not bipartite, then there is no way that a dishonest oracle can fool the nodes, and make them all accept the graph.

Interestingly, the certificates provided to the nodes by the oracle can often be computed by the nodes themselves, at low cost, during some pre-computation. For instance, a spanning tree construction algorithm is usually simply asked to encode the tree T locally at each node v , say by a pointer $p(v)$ to the parent of v in the tree. However, it is possible to ask the algorithm to also provide a distributed proof that T is a spanning tree. Such a proof may be encoded distributedly by providing each node with a certificate containing, e.g., the ID of the root of T , and the distance $d(v)$ from v to the root (see, e.g., [2, 6, 32]). Indeed, every node v but the root can simply check that $d(p(v)) = d(v) - 1$ (to guarantee the absence of cycles), and that it was given the same root-ID as all its neighbors in the network (for

89 guaranteeing the uniqueness of the tree).

90 **Distributed Interactive Protocols.**

91 The good news is that all (Turing-decidable) predicates on graphs admit a proof-labeling
92 scheme [34]. The bad news is that there are simple graph properties (e.g., existence
93 of a non-trivial automorphism [34], non 3-colorability [29], bounded diameter [10], etc.)
94 which require certificates on $\tilde{\Omega}(n)$ bits in n -node graphs. Such huge certificates do not
95 fit with the requirement that checking algorithms must not only be local, but they must
96 also consume little bandwidth. Randomized proof-labeling schemes [24] enable to limit the
97 bandwidth consumption, but this is often to the cost of increasing the space-complexity of the
98 nodes. However, motivated by cloud computing, which may provide large-scale distributed
99 systems with the ability to interact with an external party, Kol, Oshman, and Saxena [33]
100 introduced the notion of *distributed interactive protocols*. In such protocols, a centralized
101 non-trustable oracle with unlimited computation power (a.k.a. Merlin) exchanges messages
102 with a randomized distributed algorithm (a.k.a. Arthur). Specifically, Arthur and Merlin
103 perform a sequence of exchanges during which every node queries the oracle by sending a
104 random bit-string, and the oracle replies to each node by sending a bit-string called *proof*.
105 Neither the random strings nor the proofs need to be the same for each node. After a certain
106 number of rounds, every node exchange information with its neighbors in the network, and
107 decides (i.e., it outputs accept or reject). It was proved that many predicate requiring large
108 certificate whenever using proof-labeling schemes, including the existence of a non-trivial
109 automorphism, have distributed interactive protocols with proofs on $O(\log n)$ bits [33].

110 Naor, Parter, and Yogev [40] recently designed a compiler for automatically translating
111 standard centralized interactive protocols to distributed interactive protocols. In particular,
112 by using this compiler, every linear-time algorithm for deciding the membership to some
113 fixed graph class can be translated into a $\text{dMAM}(O(\log n))$ protocol, that is, a distributed
114 interactive protocol with $O(\log n)$ -bit proof size in n -node graphs, and three interactions
115 between Merlin and Arthur: Merlin provides every node with a first part of the proof,
116 on $O(\log n)$ bits, then every node challenges Merlin with a random bit-string on $O(\log n)$
117 bits, and finally Merlin replies to every node with the second part of the proof, again on
118 $O(\log n)$ bits. Every node then performs a single round of communication with its neighbors,
119 exchanging $O(\log n)$ -bit messages, and individually outputs accept or reject. As a corollary,
120 there is a $\text{dMAM}(O(\log n))$ protocol for many graph classes, including planar graphs, graphs
121 with bounded genus, graphs with bounded treewidth, etc.

122 **The Limits of Distributed Interactive Protocols.**

123 Although the compiler in [40] is quite generic and powerful, it remains that the resulting
124 interactive protocols are often based on many interactions between Merlin and Arthur. This
125 raises the question of whether there exist protocols based on fewer interactions for the
126 aforementioned classes of graphs, while keeping the proof size small (e.g., on $O(\text{polylog } n)$
127 bits). Note that, with this objective in mind, proof-labeling schemes are particularly desirable
128 as they do not require actual interactions. Indeed, as mentioned before, the certificates may
129 often be constructed a priori by the nodes themselves. Unfortunately, under the current
130 knowledge, establishing lower bounds on the number of interactions between the prover Merlin
131 and the distributed verifier Arthur, as well as lower bounds on the proof size, not to speak
132 about tradeoffs between these two complexity measures, remains challenging. Therefore, it is
133 not known whether $\text{dMAM}(O(\log n))$ protocols are the best that can be achieved for graph

134 classes such as graphs with bounded genus, or graphs with bounded treewidth.

135 **Graphs with Bounded Genus.**

136 In this paper, we focus on the class of graphs with bounded genus, for several reasons. First,
 137 this class is among the prominent representative of sparse graphs [42], and the design of fast
 138 algorithms for sparse graphs is not only of the utmost interest for centralized, but also for
 139 distributed computing (see, e.g., [3, 8, 12, 15, 26, 27, 28, 36, 37, 49]), as many real-world
 140 physical or logical networks are sparse. Second, graphs of bounded genus, including planar
 141 graphs, have attracted lots of attention recently in the distributed computing framework
 142 (see, e.g., [4, 5, 25]), and it was shown that this large class of graphs enjoys distributed
 143 exact or approximation algorithms that overcome several known lower bounds for general
 144 graphs [35, 46, 48]. Last but not least, it appears that the graph classes for which proof-
 145 labeling schemes require certificates of large size are not closed under node-deletion, which
 146 yields the question of whether every graph family closed under node-deletion (in particular
 147 graph families closed under taking minors) have proof-labeling schemes with certificates of
 148 small size. This was recently shown to be true for planar graphs [21], but the question is
 149 open beyond this class, putting aside simple classes such as bipartite graphs, forests, etc.

150 As for the class of planar graphs, and for the class of graphs with bounded genus, every
 151 graph class \mathcal{G} that is closed under taking minors has a finite set of forbidden minors. As a
 152 consequence, as established in [21], there is a simple proof-labeling scheme with $O(\log n)$ -
 153 bit certificates for $\overline{\mathcal{G}}$, i.e., for *not* being in \mathcal{G} . The scheme simply encodes a forbidden
 154 minor present in G in a distributed manner for certifying that $G \notin \mathcal{G}$. Therefore, for every
 155 $k \geq 0$, there exists a simple proof-labeling scheme with $O(\log n)$ -bit certificates for genus
 156 or non-orientable genus *at least* k . The difficulty is to design a proof-labeling scheme with
 157 $O(\log n)$ -bit certificates for genus or non-orientable genus *at most* k .

158 **1.2 Our Results**

159 **1.2.1 Compact Proof-Labeling Schemes for Graphs of Bounded Genus**

160 Recall that planar graphs are graphs embeddable on the 2-dimensional sphere S^2 (without
 161 edge-crossings). Graphs with genus 1 are embeddable on the torus \mathbb{T}_1 , and, more generally,
 162 graphs with genus $k \geq 0$ are embeddable on the closed surface \mathbb{T}_k obtained from S^2 by
 163 adding k *handles*. We show that, for every $k \geq 0$, there exists a proof-labeling scheme for
 164 the class of graphs with genus at most k , using certificates on $O(\log n)$ bits. This extends
 165 a recent proof-labeling scheme for planar graphs [21] to graphs with arbitrary genus $k \geq 0$.
 166 Note that the certificate-size of our proof-labeling schemes is optimal, in the sense there are
 167 no proof-labeling schemes using certificates on $o(\log n)$ bits, even for planarity [21]¹.

168 For every $k \geq 1$, our proof-labeling schemes also apply to the class of graphs with *non-*
 169 *orientable genus* (a.k.a., Euler genus) at most k , that is, they also hold for graphs embeddable
 170 on a *non-orientable* surface with genus k . Graphs with non-orientable k are indeed graphs
 171 embeddable on the closed surface \mathbb{P}_k obtained from S^2 by adding k *cross-caps*. **Some more**
 172 **precise definitions and descriptions are given later, in Section 2.1.**

173 This paper therefore demonstrates that the ability of designing proof-labeling schemes
 174 with small certificates for planar graphs is not a coincidental byproduct of planarity, but

¹ The goal of this paper is not to optimize the size of the certificates as a function of the genus k , but it is not hard to see that our certificates have size $O(2^k \log n)$.

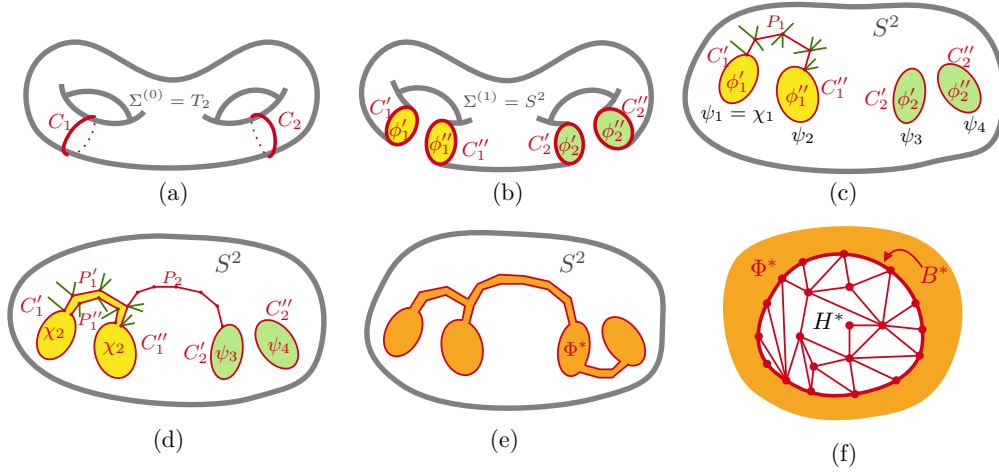


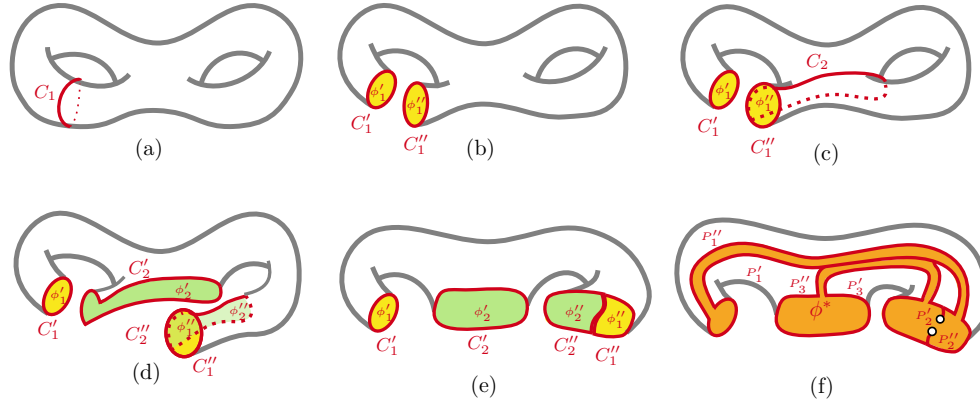
Figure 1 An idealistic scenario where a graph G embedded on \mathbb{T}_2 has disjoint non-separating cycles. In this case, we cut along two disjoint non-separating cycles C_1 and C_2 , get four supplementary faces, and merge these faces by cutting along disjoint paths, until we get only one face (which is orange in the drawing). Let us give more details, using the notations of the proof. The supplementary faces are $\phi'_1, \phi''_1, \phi'_2$ and ϕ''_2 , and they are renamed ψ_1, ψ_2, ψ_3 and ψ_4 respectively, for convenience. These faces are merged step by step: a duplicated path is used to merge such a face ψ_i with the face χ_{i-1} originating from the merge of the faces ψ_1 to ψ_{i-1} (with $\chi_1 = \psi_1$). Picture (d) illustrates the merge of χ_2 with ψ_3 . In the end, we have a single merged face that we call Φ^* . The latter face Φ^* can be seen as the infinite face of the planar graph embedding obtained by these transformations.

175 this ability extends to much wider classes of sparse graphs closed under taking minors. This
 176 provides hints that proof-labeling schemes with small certificates can also be designed for
 177 very many (if not all) natural classes of sparse graphs closed under vertex-deletion.

1.2.2 Our Techniques

179 Our proof-labeling schemes are obtained thanks to a local encoding of a mechanism enabling
 180 to “unfold” a graph G of genus or non-orientable genus k into a planar graph \widehat{G} , by a
 181 series of vertex-duplications. Specifically, for graphs of genus k , i.e., embeddable on an
 182 orientable surface \mathbb{T}_k , we construct a sequences $G^{(0)}, \dots, G^{(k)}$ where $G^{(0)} = G$, $G^{(k)} = \widehat{G}$,
 183 and, for every $i = 0, \dots, k$, $G^{(i)}$ has genus $k - i$. For $i \geq 1$, the graph $G^{(i)}$ is obtained from
 184 $G^{(i-1)}$ by identifying a non-separating cycle C_i in $G^{(i-1)}$, and duplicating the vertices and
 185 cycles of C_i (see Figure 1(a-b)). (Recall that a non-separating cycle, is a cycle that can be
 186 removed without making the graph disconnected.) This enables to “cut” a handle of the
 187 surface \mathbb{T}_{k-i+1} , resulting in a closed surface \mathbb{T}_{k-i} with genus one less than \mathbb{T}_{k-i+1} , while
 188 the embedding of $G^{(i-1)}$ on \mathbb{T}_{k-i+1} induces an embedding of $G^{(i)}$ on \mathbb{T}_{k-i} . The graph \widehat{G} is
 189 planar, and has $2k$ special faces $\phi'_1, \phi''_1, \dots, \phi'_k, \phi''_k$, where, for $i = 0, \dots, k$, the faces ϕ'_i and
 190 ϕ''_i results from the duplication of the face C_i (see Figure 1(c)).

191 The proof-labeling scheme needs to certify not only the planarity of \widehat{G} , but also the
 192 existence of the faces $\phi'_1, \phi''_1, \dots, \phi'_k, \phi''_k$, and a proof that they are indeed faces, which is
 193 non-trivial. Therefore, instead of keeping the $2k$ faces as such, we connect them by a sequence
 194 of paths P_1, \dots, P_{2k-1} . By duplicating each path P_i into P'_i and P''_i , the two faces χ and
 195 ψ connected by a path P_i is transformed into a single face, while planarity is preserved.
 196 Intuitively, the new face is the “union” of χ , ψ , and the “piece in between” P'_i and P''_i (see
 197 Figure 1(d)). The whole process eventually results in a planar graph H with a single special



■ **Figure 2** A more complex unfolding a graph G embedded on \mathbb{T}_2 . Faces created by duplications have not disjoint boundaries, and parts of previous duplicated paths are used to create new paths.

198 face ϕ (see Figure 1(e-f)). In fact, the paths P_i , $i = 1, \dots, 2k - 1$ do not only serve the
 199 objective of merging the $2k$ faces $\phi'_1, \phi''_1, \dots, \phi'_k, \phi''_k$ into a single face ϕ , but also serve the
 200 objective of keeping track of consistent orientations of the boundaries of these faces. The
 201 purpose of these orientations is to provide the nodes with the ability to locally check that
 202 the $2k$ faces can indeed be paired for forming k handles.

203 The planarity of H and the existence of the special face ϕ can be certified by a slight
 204 adaptation of the proof-labeling scheme for planarity in [21]. It then remains to encode the
 205 sequence of cycle and path-duplications locally, so that every node can roll back the entire
 206 process, for identifying the cycles C_i , $i = 1, \dots, k$, and the paths P_j , $j = 1, \dots, 2k - 1$, and
 207 for checking their correctness.

208 Among many issues, a very delicate problem is that, as opposed to cycles and paths drawn
 209 on a surface, which can be chosen to intersect at few points, these cycles and paths are in
 210 graphs embedded on surfaces, and thus they may intersect a lot, by sharing vertices or even
 211 edges. Figure 1 displays an idealistic scenario in which the cycles C_i 's are disjoint, the paths
 212 P_j 's are disjoint, and these cycles and paths are also disjoint. However, this does not need to
 213 be the case, and the considered cycles and paths may mutually intersect in a very intricate
 214 manner. For instance, Figure 2 displays a case in which C_2 intersects with C_1'' , P_2' and P_2''
 215 are reduced to single vertices, and P_3'' intersects with P_1' . It follows that the sequence of
 216 duplications may actually be quite cumbersome in general, with some nodes duplicated many
 217 times. As a consequence, keeping track of the boundaries of the faces is challenging, especially
 218 under the constraint that all information must be distributed, and stored at each node using
 219 $O(\log n)$ bits only. Also, one needs to preserve specific orientations of the boundaries of
 220 the faces, for making sure that not only the two faces ϕ'_i and ϕ''_i corresponding to a same
 221 cycle C_i can be identified, but also that they can be glued together appropriately in a way
 222 resulting to a handle, and not be glued like, e.g., a Klein bottle.

223 The case of graphs embedded on a non-orientable closed surface causes other problems,
 224 including the local encoding of the cross-caps, and the fact that decreasing the genus of a
 225 non-orientable closed surface by removing a cross-cap may actually result in a closed surface
 226 that is orientable. Indeed, eliminating cross-caps is based on *doubling* a non-orientable cycle
 227 of the graph, and this operation may result in a graph embedded on a surface that is actually
 228 orientable. (This phenomenon did not arise in the case of orientable surfaces, as removing a
 229 handle from an orientable closed surface by cycle-duplication results in a graph embedded

230 on an orientable closed surface.) Thus, the proof-labeling scheme for bounded non-orientable
231 genus has to encode not only the identification the cross-caps, but also of faces to be identified
232 for forming handles.

233 For guaranteeing certificates on $O(\log n)$ bits, our proof-labeling schemes distribute the
234 information evenly to the certificates provided to the nodes, using the fact that graphs
235 of bounded (orientable or non-orientable) genus have bounded *degeneracy*. This property
236 enables to store certificates on $O(\log n)$ bits at each node, even for nodes that have arbitrarily
237 large degrees.

238 We complete this brief summary of our techniques with two remarks.

239 **On the cuts.** Modifying a graph of bounded genus by performing a sequence of cuts for
240 eventually producing a planar graph has already been used in the literature — see, e.g.,
241 [31], where a probabilistic embedding of bounded genus graphs into planar graphs is
242 designed. However, using this techniques in the framework of distributed computing is,
243 to our knowledge, new, and poses additional challenges. In particular, dealing with the
244 intersections of the aforementioned paths and cycles is not much difficult in centralized
245 computing (typically, few virtual nodes may be added, or the graph may even be
246 triangulated, for avoiding intersections), but the techniques used in the centralized setting
247 do not carry over easily to the decentralized setting. More generally, the fact that every
248 step of the transformation of a graph of bounded (non-orientable) genus into a planar
249 graph must be verifiable locally in a distributed manner imposes strong constraints, and
250 restricts the set of techniques that can be used. As a consequence, we could not pick a
251 transformation from the shelf for using it as a black box, but we had to come up with a
252 specific one, bearing close similarities with existing ones, for carefully monitoring every
253 step of it, and checking the ability to implement this step in a distributed manner using
254 small certificates.

255 **On the general approach.** An approach conceptually simpler than the one used in this
256 paper would have been to use induction, simply assuming the existence of a proof-labeling
257 scheme with $O(\log n)$ -bit certificates for graphs with (non-orientable) genus $k \geq 0$, and
258 then constructing a proof-labeling scheme with $O(\log n)$ -bit certificates for graphs with
259 (non-orientable) genus $k + 1$. However, although we are not claiming that such a desirable
260 and conceptually simpler approach is impossible to use, we strongly believe that it may
261 simply be not the right approach, for at least two reasons. First, the proof-labeling scheme
262 in [21] certifies planarity, but it does not provide a way to certify all the faces of a planar
263 embedding. Indeed, it only provides a certification for a single specific face, namely the
264 outer-face. Given an arbitrary cycle in the graph, certifying that this cycle corresponds
265 to the boundary of a face in the planar embedding is not provided in [21], and the design
266 of a compact proof-labeling scheme for this property appears to be non trivial. Note in
267 particular that we do not assume that the graphs we manipulate have unique embeddings,
268 thus a cycle might be a face in one embedding, but not in another embedding. Second,
269 even if the previous problem could be solved, it would remain that the orientations of the
270 pair of faces to be merged at each level of the induction are crucial. One needs to make
271 sure that the nodes can locally check that the orientations provided by the non-trustable
272 prover are correct, for distinguishing handles from cross-caps. The inductive design of a
273 compact proof-labeling scheme for this property appears to be even more challenging.
274 These two issues are serious obstacles to the development of an inductive construction,
275 and, to overcome them, we adopted the approach consisting to “unwrap” the whole
276 construction. This is less elegant and comprehensible than an inductive construction,
277 but this allowed us to (1) identify a single face in the planar embedding, which can be

278 certified (e.g., using [21]), and (2) provide an orientation to the boundary of this face, for
 279 enabling to certify that the orientation given by the non-trustable prover to each and
 280 every pair of faces to be merged is indeed correct.

281 1.3 Related Work

282 Bounded-degree graphs form one of the most popular class of sparse graphs studied in
 283 the context of design and analysis of distributed algorithms, as witnessed by the large
 284 literature (see, e.g., [45]) dedicated to construct *locally checkable labelings* (e.g., vertex
 285 colorings, maximal independent sets, etc.) initiated a quarter of a century ago by the seminal
 286 work in [41]. Since then, other classes of sparse graphs have received a lot of attention,
 287 including planar graphs, and graphs of bounded genus. In particular, there is a long history
 288 of designing distributed approximation algorithms for these classes, exemplified by the case
 289 of the minimum dominating set problem. One of the earliest result for this latter problem is
 290 the design of a constant-factor approximation algorithm for planar graphs, performing in a
 291 constant number of rounds [36]. This result is in striking contrast with the fact that even
 292 a poly-logarithmic approximation requires at least $\Omega(\sqrt{\log n / \log \log n})$ rounds in arbitrary
 293 n -node networks [35]. The paper [36] has paved the way for a series of works, either improving
 294 on the complexity and the approximation ratio [15, 37, 49], or using weaker models [50], or
 295 tackling more general problems [13, 14], or proving lower bounds [30, 15]. The minimum
 296 dominating set problem has then been studied in more general classes such as graphs with
 297 bounded arboricity [37], minor-closed graphs [12], and graphs with bounded expansion [3].
 298 Specifically, for graphs with bounded genus, it has been shown that a constant approximation
 299 can be obtained in time $O(k)$ for graphs of genus k [4], and a $(1 + \epsilon)$ -approximation algorithm
 300 has recently been designed, performing in time $O(\log^* n)$ [5].

301 Several other problems, such as maximal independent set, maximal matching, etc., have
 302 been studied for the aforementioned graph classes, and we refer to [18] for an extended
 303 bibliography. In addition to the aforementioned results, mostly dealing with local algorithms,
 304 there are recent results in computational models taking into account limited link bandwidth,
 305 for graphs that can be embedded on surfaces. For instance, it was shown that a combinatorial
 306 planar embedding can be computed efficiently in the CONGEST model [26]. Such an
 307 embedding can then be used to derive more efficient algorithms for minimum-weight spanning
 308 tree, min-cut, and depth-first search tree constructions [27, 28]. Finally, it is worth mentioning
 309 that, in addition to algorithms, distributed data structures have been designed for graphs
 310 embedded on surfaces, including a recent optimal adjacency-labeling for planar graphs [8, 16],
 311 and routing tables for graphs of bounded genus [25] as well as for graphs excluding a fixed
 312 minor [1].

313 Proof-labeling schemes (PLS) were introduced in [34], and different variants were later
 314 introduced. Stronger forms of PLS include *locally checkable proofs* (LCP) [29] in which
 315 nodes forge their decisions on the certificates and on the whole states of their neighbors, and
 316 t -PLS [20] in which nodes perform communication at distance $t \geq 1$ before deciding. Weaker
 317 forms of PLS include *non-deterministic local decision* (NLD) [22] in which the certificates
 318 must be independent from the identity-assignment to the nodes. PLS were also extended
 319 by allowing the verifier to be randomized (see [24]). Such protocols were originally referred
 320 to as *randomized PLS* (RPLS), but are nowadays referred to as distributed Merlin-Arthur
 321 (dMA) protocols.

322 The same way NP is extended to the complexity classes forming the Polynomial Hierarchy,
 323 by alternating quantifiers, PLS were extended to a hierarchy of distributed decision classes [7,
 324 19], which can be viewed as resulting from a game between a prover and a *disprover*.

325 Recently, *distributed interactive proofs* were formalized [33], and the classes $\text{dAM}[k](f(n))$
 326 and $\text{dMA}[k](f(n))$ were defined, where $k \geq 1$ denotes the number of alternations between
 327 the centralized Merlin and the decentralized Arthur, and $f(n)$ denotes the size of the proof
 328 — $\text{dAM}[3](f(n))$ is also referred to as $\text{dMAM}(f(n))$. Distributed interactive protocols for
 329 problems like the existence of a non-trivial automorphism (AUT), and non-isomorphism (ISO)
 330 were designed and analyzed in [33]. The follow up paper [40] improved the complexity of
 331 some of the protocols in [33], either in terms of the number of interactions between the prover
 332 and the verifier, and/or in terms of the size of the certificates. A sophisticated generic way
 333 for constructing distributed IP protocols based on sequential IP protocols is presented in [40].
 334 One of the main outcome of this latter construction is a dMAM protocol using certificates
 335 on $O(\log n)$ bits for all graph classes whose membership can be decided in linear time. For
 336 other recent results on distributed interactive proof, see [11, 23].

337 It is worth noticing that a very recent arXiv paper [17] provides an alternative proof of the
 338 results of this paper, by certifying (i) the faces of the embedding using the Heffter-Edmonds-
 339 Ringel rotation principle, and (ii) the genus of the embedding using Euler’s Formula.

340 1.4 Organization of the Paper

341 The next section provides the reader with basic notions regarding graphs embedded on closed
 342 surfaces, and formally defines our problem. Section 3 describes how to “unfold” a graph G
 343 of genus k , for producing a planar graph H with a special face ϕ . The section also describes
 344 how, given a planar graph H with a special face ϕ , one can check that (H, ϕ) results from the
 345 unfolding of a graph G with genus k . Then, Section 4 presents our first main result, that is, a
 346 proof-labeling scheme for the class of graphs with bounded genus. In particular, it describes
 347 how to encode the description of the pair (H, ϕ) from Section 3, and, more importantly, how
 348 to locally encode the whole unfolding process in a distributed manner, using certificates on
 349 $O(\log n)$ bits, which allow the nodes to collectively check that their certificates form a proof
 350 that G has genus k . Section 5 presents our second main result, by showing how to extend the
 351 proof-labeling scheme of Section 4 to the class of graphs with bounded non-orientable genus.
 352 Finally, Section 6 concludes the paper with a discussion about the obstacles to be overcome
 353 for the design of a proof-labeling scheme for the class of graphs excluding a fixed minor.

354 2 Definitions, and Formal Statement of the Problem

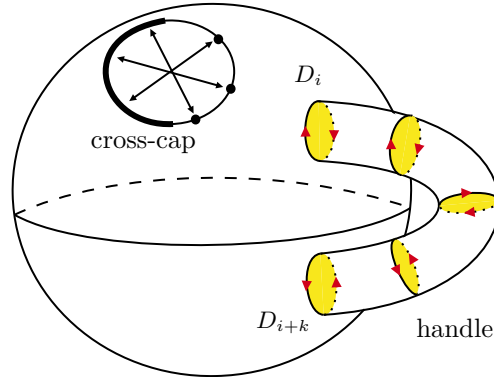
355 This section contains a brief introduction to graphs embedded on surfaces, and provides the
 356 formal statement of our problem.

357 2.1 Closed Surfaces

358 Most of the notions mentioned in this section are standard, and we refer to, e.g., Massey et
 359 al. [38] for more details.

360 2.1.1 Definition

361 Recall that a *topological space* is a pair (X, T) where X is a set, and T is a topology on X
 362 (e.g., T is a collection of subsets of X , whose elements are called *open sets*, **satisfying the**
 363 **following properties : the set X and the empty set are open, any finite intersection of open**
 364 **sets is an open set, and any arbitrary union of open sets is an open set**). A topological space
 365 may be denoted by X if there is no ambiguity about the topology on X . **Also recall that**



■ **Figure 3** Handles and cross-cap.

366 a topological space X is *compact* if, from any set of open sets whose union is X , one can
 367 extract a finite set of open sets whose union is finite. A function $f : X \rightarrow Y$ between two
 368 topological spaces is *continuous* if the inverse image of every open set in Y is open in X . A
 369 *homeomorphism* is a bijection that is continuous, and whose inverse is also continuous. A
 370 *topological path* in X is a continuous function $P : [0, 1] \rightarrow X$. The space X is *path-connected*
 371 if for any pairs x, y of points of X , there exists a topological path P such that $P(0) = x$ and
 372 $P(1) = y$.

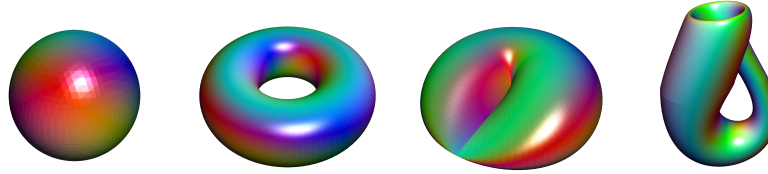
373 ► **Definition 1.** A closed surface Σ is a path-connected², compact space that is locally
 374 homeomorphic to a disk of \mathbb{R}^2 , (i.e. for each $x \in \Sigma$, there exists an open set S_x containing x
 375 such that S_x is homeomorphic to an closed disk of \mathbb{R}^2 , the topology T_{S_x} used for S_x being the
 376 set $T_{S_x} = \{S \cap S_x, S \in T\}$).

377 2.1.2 Construction

378 Some closed surfaces can be obtained by the following construction. Let S^2 be the 2-
 379 dimensional sphere. For $k \geq 0$, given $2k$ disks D_1, D_2, \dots, D_{2k} on the surface of S^2 , with
 380 pairwise disjoint interiors, let us direct clockwise the boundaries of D_1, \dots, D_k , and let us
 381 direct counterclockwise the boundaries of D_{k+1}, \dots, D_{2k} . Next, let us remove the interior of
 382 each disk, and, for $1 \leq i \leq k$, let us identify (i.e., glue) the boundary of D_i with the boundary
 383 D_{i+k} in such a way that directions coincide (see Figure 3). The resulting topological space is
 384 denoted by \mathbb{T}_k . In particular, \mathbb{T}_1 is the torus, and $\mathbb{T}_0 = S^2$. For every i , identifying D_i and
 385 D_{i+k} results in a *handle*. It follows that \mathbb{T}_k contains k handles.

386 Another family of closed surfaces is constructed as follows. Let D_1, \dots, D_k be $k \geq 1$ disks
 387 with pairwise disjoint interiors. Let us again remove the interior of each disk. For every
 388 $1 \leq i \leq k$, and for every antipodal point v and v' of the boundary of D_i , let us identify (i.e.,
 389 glue) the points v and v' (see Figure 3). The resulting topological space is denoted by \mathbb{P}_k .
 390 In particular, \mathbb{P}_1 is the projective plane, and \mathbb{P}_2 is the Klein bottle (\mathbb{P}_0 is not defined). For
 391 every i , the operation performed on D_i results in a *cross-cap*. It follows that \mathbb{P}_k contains k
 392 cross-caps.

² Path-connected can actually be replaced by connected (i.e., cannot be partitioned in two open sets) here, because, under the hypothesis of local homeomorphy to a disk, the notions of path-connectivity and connectivity are equivalent.



■ **Figure 4** The sphere, torus, projective plane, and Klein Bottle.

393 The surfaces resulting from the above constructions can thus be *orientable* (e.g., the
 394 sphere \mathbb{T}_0 or the torus \mathbb{T}_1) or not (e.g., the projective plane \mathbb{P}_1 or the Klein Bottle \mathbb{P}_2), as
 395 displayed on Figure 4.

396 2.1.3 Orientability

397 For defining orientability of a closed surface Σ , we use the notion of *curve*, defined as a
 398 continuous function $C : S^1 \rightarrow \Sigma$, where S^1 denotes the unidimensional sphere (homeomorphic
 399 to, e.g., the trigonometric circle). A curve is *simple* if it is injective. A simple curve C is
 400 *orientable* if one can define the left side and the right side of the curve at every point of the
 401 curve in a consistent manner. Specifically, a curve C is orientable if, for every $x \in C$, there
 402 exists a neighborhood N_x of x such that $N_x \setminus C$ has two connected components, one called
 403 the left side $L(N_x)$ of N_x , and the other the right side $R(N_x)$ of N_x , such that, for every
 404 $x, x' \in C$ and every $y \in \Sigma$,

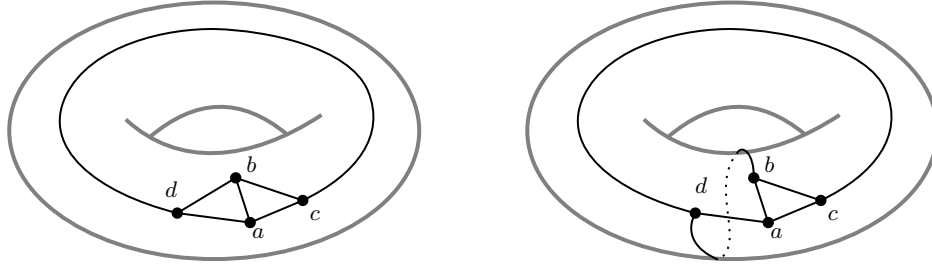
$$405 \quad (y \in N_x \cap N_{x'}) \wedge (y \in L(N_x)) \implies y \in L(N_{x'}).$$

406 A closed surface Σ is orientable if every simple curve of X is orientable. It is easy to check
 407 that orientability is a topological invariant. That is, if Σ and Σ' are two homeomorphic
 408 topological spaces, then Σ is orientable if and only if Σ' is orientable.

409 2.1.4 Genus of a Surface

410 An orientable closed surface Σ is of *genus* k if it is homeomorphic to a closed surface \mathbb{T}_k
 411 constructed as in Section 2.1.2. The Classification Theorem of orientable closed surfaces (see,
 412 e.g., [9]) states that every orientable closed surface has a genus. That is, for every orientable
 413 surface Σ , there exists a unique $k \geq 0$ such that Σ is of genus k . The fact that every pair of
 414 orientable closed surfaces with the same genus k are homeomorphic, justifies that a unique
 415 notation can be adopted for these surfaces, and any orientable closed surface of genus k is
 416 denoted by \mathbb{T}_k . Observe however that two closed surfaces that are homeomorphic are not
 417 necessarily homotopic, i.e., they may not be continuously deformable into each other (for
 418 instance, the torus is not homotopic to the trefoil knot, although both are homeomorphic).

419 The genus can also be defined for non-orientable closed surfaces. For $k \geq 1$, a non-
 420 orientable closed surface is said to be of *genus* k if it is homeomorphic to a closed surface \mathbb{P}_k
 421 constructed as in Section 2.1.2. Again, the Classification Theorem of non-orientable closed
 422 surfaces (see, e.g., [9]) states that every non-orientable closed surface has a genus. That
 423 is, for every non-orientable closed surface Σ , there exists a unique $k \geq 0$ such that Σ is of
 424 genus k . As for orientable surfaces, every pair of non-orientable closed surfaces of genus k
 425 are homeomorphic, and a non-orientable closed surface of genus k is denoted by \mathbb{P}_k .



■ **Figure 5** Two embeddings of K_4 on the torus \mathbb{T}_1 . The one on the left is not a 2-cell embedding since the non-triangular face is not homeomorphic to a closed disk. This situation can occur because K_4 is of genus 0, not 1 (see Lemma 3). The embedding on right is a 2-cell embedding.

2.2 Graphs Embedded on Surfaces

In this section, we recall standard notions related to graph embeddings on surfaces, and we refer to Mohar and Thomassen [39] for more details. Throughout the paper, all considered graphs are supposed to be simple (no multiple edges, and no self-loops), and connected.

2.2.1 Topological Embeddings

Given a graph $G = (V, E)$, and a closed surface Σ , a *topological embedding* of G on Σ is given by (1) an injective mapping $f : V \rightarrow \Sigma$, and, (2) a topological path $f_e : [0, 1] \rightarrow \Sigma$ defined for every edge e such that:

- if $e = \{v, v'\} \in E$, then $f_e(\{0, 1\}) = \{f(v), f(v')\}$, and
- if $e, e' \in E$ and $e \neq e'$, then $f_e([0, 1]) \cap f_{e'}([0, 1]) = \emptyset$.

The second condition is often referred to as the *non-crossing* condition. See Figure 5 for two embeddings of the complete graph K_4 on \mathbb{T}_1 . Throughout the paper, we may identify a vertex v with its representation $f(v)$, and an edge e with its representation f_e (i.e., the image $f_e([0, 1])$ of $[0, 1]$ by f_e), even referred to as $f(e)$ in the following. The set $\cup_{e \in E} f(e)$ is called the *skeleton* of the embedding, and is denoted by $\text{Sk}(G)$. Each connected component of $\Sigma \setminus \text{Sk}(G)$ is an open set of Σ (as complement of a closed set), called a *face* of the embedding. In fact, in this paper, we will abuse notation, and often refer to G instead of $\text{Sk}(G)$ when referring to the embedding of G on Σ .

2.2.2 2-Cell Embeddings

We now recall a slightly more sophisticated, but significantly richer form of topological embedding, called *2-cell embedding*. A 2-cell embedding is a topological embedding such that every face is homeomorphic to an open disk of \mathbb{R}^2 .

In a 2-cell embedding of a graph G , the border of a face can be described by giving a so-called *boundary (closed) walk*, that is, an ordered list (v_0, \dots, v_r) of non-necessarily distinct vertices of G , where, for $i = 0, \dots, r - 1$, $\{v_i, v_{i+1}\} \in E(G)$, and $\{v_r, v_0\} \in E(G)$. The vertices and edges of a face are the images by the embedding of the vertices and edges of the boundary walk. The boundary walk is however not necessarily a simple cycle, as an edge may appear twice in the walk, once for each direction, and a vertex may even appear many times.

For instance, Figure 5 displays two embeddings of the complete graph K_4 on the torus \mathbb{T}_1 . The embedding on the left is not a 2-cell embedding. Indeed, this embedding results in three faces, including the two faces with boundary walk (a, b, c) and (a, b, d) . The third

458 face is however not homeomorphic to an open disk (there is a hole in it, resulting from the
 459 hole in the torus). On the other hand, the embedding on the right in Figure 5 is a 2-cell
 460 embedding. Indeed, there are two faces, including the face with boundary walk (a, b, c) . The
 461 other face is also homeomorphic to an open disk. A boundary walk of this latter face is
 462 $(d, a, b, d, c, a, d, b, c)$. This can be seen by starting from d , traversing the edge $\{d, a\}$, and
 463 adopting the “left-hand rule” when entering a vertex, leading from a to b , then back to d ,
 464 next to c , etc. Notice that this boundary walk uses some edges twice. It follows that the
 465 closure of a face is not necessarily homeomorphic to a closed disk, even in a 2-cell embedding.

466 We complete the section with an observation, which allows us to restrict our attention
 467 to cycles in graphs instead of arbitrary curves in topological spaces. It also illustrates the
 468 interest of 2-cell embeddings (the result does not necessarily hold for arbitrary embeddings,
 469 as illustrated by the embedding on the left of Figure 5). In the following, *contractible* means
 470 homotopic to a point.

471 ► **Lemma 2.** *For every graph G , and every closed surface Σ , any 2-cell embedding of G on*
 472 *Σ satisfies that every closed curve in Σ is either contractible, or homotopic to a closed cycle*
 473 *of $\text{Sk}(G)$.*

474 The rough reason why the result holds is that, in a 2-cell embedding, any sub-path of a
 475 path traversing a face can be replaced by a sub-path following the border of the face. (This
 476 is not necessarily true for a general embedding).

477 2.2.3 Genus and non-orientable genus of a Graph

478 For any graph G , there exists $k \geq 0$ such that G can be embedded on \mathbb{T}_k , as any embedding
 479 of G in the plane with x pairs of crossing edges induces an embedding of G on \mathbb{T}_x without
 480 crossings, by replacing each crossing with a handle. Also, if G can be embedded on \mathbb{T}_k , then
 481 G can be embedded on $\mathbb{T}_{k'}$ for every $k' \geq k$. The *genus* of a graph G is the smallest k such
 482 that there exists an embedding of G on \mathbb{T}_k . Similarly, the *non-orientable genus*, or *Euler*
 483 *genus* of G , is defined as the smallest k such that there exists an embedding of G on \mathbb{P}_k .

484 The embeddings of graphs of genus k on \mathbb{T}_k have a remarkable property (see, e.g., [51]).

485 ► **Lemma 3.** *Every embedding of a graph G of genus k on \mathbb{T}_k is a 2-cell embedding.*

486 The same property does not necessarily hold for graphs with bounded non-orientable
 487 genus. However, some weaker form of Lemma 3 can be established (see, e.g., [44]).

488 ► **Lemma 4.** *For every graph G of non-orientable genus k , there exists a 2-cell embedding*
 489 *of G on \mathbb{P}_k .*

490 The next result is extremely helpful for computing the genus of a graph, and is often
 491 referred to as the Euler-Poincaré formula [47].

492 ► **Lemma 5.** *Let $G = (V, E)$, and let Σ be a closed surface of genus k . Let us consider any*
 493 *2-cell embedding of G on Σ , and let F be the set of faces of this embedding. If Σ is orientable*
 494 *then $|V| - |E| + |F| = 2 - 2k$. If Σ is non orientable then $|V| - |E| + |F| = 2 - k$.*

495 Recall that, for $d \geq 0$, a graph G is *d-degenerate* if every subgraph of G has a node
 496 of degree at most d . Degeneracy will play a crucial role later in the paper, for evenly
 497 distributing the information to be stored in the certificates according to our proof-labeling
 498 schemes. **Graphs with bounded genus have bounded degeneracy (see, e.g., [39] Theorem 8. 3.**
 499 **1, this result is due to Heawood), as recalled below for further references.**

500 ► **Lemma 6.** *For every $k \geq 0$, every graph of genus at most k is d -degenerate with $d =$
 501 $\max(5, \frac{5+\sqrt{1+48k}}{2})$.
 502 *For every $k \geq 1$, every graph of non-orientable genus at most k is d -degenerate with
 503 $d = \max(5, \frac{5+\sqrt{1+24k}}{2})$.**

504 2.3 Formal Statement of the Problem

505 Proof-Labeling Schemes (PLS) are distributed mechanisms for verifying graph properties.
 506 More precisely, let \mathcal{G} be a graph family. A PLS for \mathcal{G} is defined as a prover-verifier pair
 507 (\mathbf{p}, \mathbf{v}) , bounded to satisfy the following. Given any graph $G = (V, E)$ whose n vertices are
 508 arbitrarily labeled by n distinct identifiers (ID) picked from a set $\{1, \dots, n^k\}$, $k \geq 1$, of
 509 polynomial range, the prover \mathbf{p} is a non-trustable oracle that provides every vertex $v \in V$
 510 with a *certificate* $c(v)$. The verifier \mathbf{v} is a distributed protocol performing a single round in
 511 parallel at all vertices, as follows. Every vertex collects the certificates of all its neighbors,
 512 and must output “accept” or “reject”, on the basis of its ID, its certificate, and the certificates
 513 of its neighbors. The pair (\mathbf{p}, \mathbf{v}) is a correct PLS for \mathcal{G} if the following two conditions hold.

514 **Completeness:** For every $G \in \mathcal{G}$, and for every ID-assignment to the vertices of G , the
 515 (non-trustable) prover \mathbf{p} can assign certificates to the vertices such that the verifier \mathbf{v}
 516 *accepts* at all vertices;

517 **Soundness:** For every $G \notin \mathcal{G}$, for every ID-assignment to the vertices of G , and for every
 518 certificate-assignment to the vertices by the non-trustable prover \mathbf{p} , the verifier \mathbf{v} *rejects*
 519 in at least one vertex.

520 The main *complexity measure* for a PLS is the size of the certificates assigned to the
 521 vertices by the prover. The objective of the paper is to design schemes with logarithmic-size
 522 certificates, for two classes of graphs: the class \mathcal{G}_k^+ , $k \geq 0$, of graphs embeddable on an
 523 orientable closed surface of genus at most k (i.e., the graphs of genus $\leq k$), and the class \mathcal{G}_k^- ,
 524 $k \geq 1$, of graphs embeddable on a non-orientable closed surface of genus at most k (i.e., the
 525 graphs of non-orientable genus $\leq k$).

526 Remark.

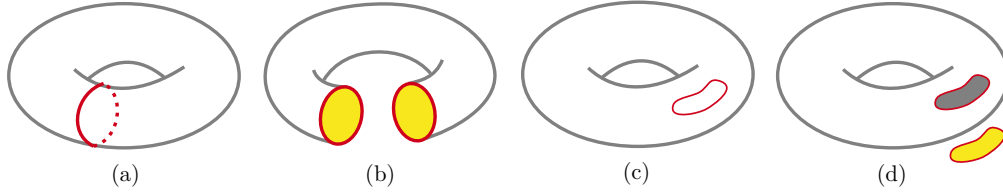
527 Throughout the rest of the paper, for $G \in \mathcal{G}_k^+$ (resp., $G \in \mathcal{G}_k^-$) with genus $k' < k$ (resp.,
 528 non-orientable genus $k' < k$), our proof-labeling scheme certifies an embedding of G on $\mathbb{T}_{k'}$
 529 (resp., on $\mathbb{P}_{k'}$). Therefore, in the following, k is supposed to denote the exact genus of G .

530 3 Unfolding a Surface

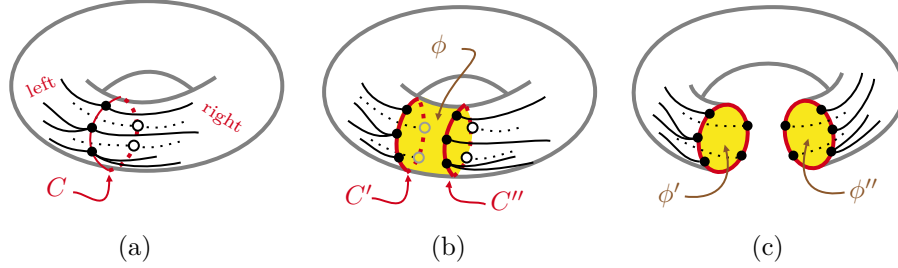
531 In this section, we describe how to “flat down” a surface, by reducing it to a disk whose
 532 boundary has a specific form. This operation is central for constructing the distributed
 533 certificates in our proof-labeling scheme. In fact, it provides a centralized certificate for
 534 bounded genus. The section is dedicated to orientable surfaces, and the case of non-orientable
 535 surfaces will be treated further in the text.

536 3.1 Separation and Duplication

537 Given a 2-cell embedding of a graph G on a closed surface Σ , a *non-separating cycle* of the
 538 embedding is a simple cycle C in G such that $\Sigma \setminus C$ is connected. Figure 6 illustrates this
 539 notion: the cycle displayed on (a) is non-separating, as shown on (b); instead, the cycle



■ **Figure 6** Separating and non-separating cycles.



■ **Figure 7** Cycle-duplication and the associated surface.

540 displayed on (c) is separating, as shown on (d). The result hereafter is a classic result, whose
 541 proof can be found in, e.g., [39, 43].

542 ► **Lemma 7.** *Let G be a graph embeddable on a closed orientable surface Σ with genus $k \geq 1$.
 543 For any 2-cell embedding of G on Σ , there exists a non-separating cycle C in G .*

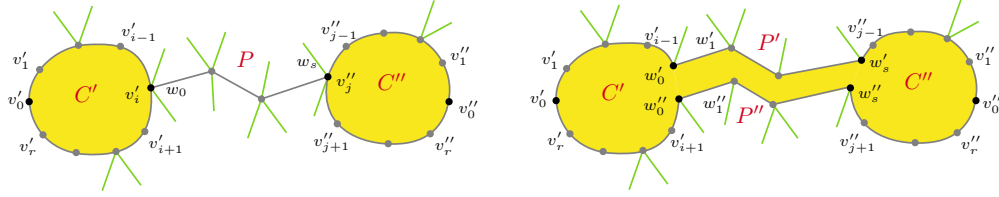
544 Note that the hypothesis that there is a 2-cell embedding is crucial: a tree can be
 545 embedded on any surface, but has no cycle.

546 3.1.1 Cycle-Duplication

547 Let G be a graph embeddable on a closed orientable surface Σ . An orientable cycle is a
 548 cycle of G whose embedding on Σ yields an orientable curve. Given a 2-cell embedding f
 549 of G on Σ , let C be a non-separating orientable cycle of G whose existence is guaranteed by
 550 Lemma 7. By definition, the left and right sides of C can be defined on the neighborhood of
 551 C . We denote by G_C the graph obtained by the *duplication* of C in G . Specifically, let us
 552 assume that $C = (v_0, \dots, v_r)$. Every vertex $w \notin C$ remains in G_C , as well as every edge non
 553 incident to a vertex of C . Every vertex v_i of C is replaced by a *left* vertex v'_i and a *right*
 554 vertex v''_i . For every $i = 0, \dots, r - 1$, $\{v'_i, v'_{i+1}\}$ and $\{v''_i, v''_{i+1}\}$ are edges of G_C , as well as
 555 $\{v'_r, v'_0\}$ and $\{v''_r, v''_0\}$. Finally, for every $i = 0, \dots, r$, and every neighbor $w \notin C$ of v_i in G , if
 556 $f(\{v_i, w\})$ meets the left of C , then $\{v'_i, w\}$ is an edge of G_C , otherwise $\{v''_i, w\}$ is an edge of
 557 G_C . The embedding f of G on Σ directly induces an embedding of G_C on Σ . Figure 7(a-b)
 558 illustrates the operation of duplication, and the resulting embedding on Σ .

559 The embedding of G_C on Σ is however not a 2-cell embedding, as it contains the face ϕ
 560 between C' and C'' on Σ , where $C' = (v'_0, \dots, v'_r)$ and $C'' = (v''_0, \dots, v''_r)$ (see Figure 7(b)).
 561 Formally, ϕ is the face with boundaries C' and C'' , and, as such, it is not homeomorphic
 562 to a disc. Let Σ_C be the closed surface³ obtained from Σ by removing ϕ , and by replacing

³ Notice that $X \setminus \phi$, $\overline{\phi'} = \phi' \cup C'$ (where $\overline{\phi'}$ denotes the adherence of ϕ'), and $\overline{\phi''} = \phi'' \cup C''$ are compact sets. Thus Σ_C is compact as the union of these three sets.



■ **Figure 8** Path-duplication.

563 ϕ with two faces ϕ' and ϕ'' with boundary walks C' and C'' , respectively (see Figure 7(c)).
 564 The embedding f of G on Σ induces a 2-cell embedding f_C of G_C on Σ_C . Also, since C is a
 565 non-separating cycle of G in Σ , the surface Σ_C is path-connected, which ensures that G_C is
 566 connected using Lemma 2.

567 Moreover, as Σ is orientable, Σ_C is also orientable. Indeed, every simple cycle of Σ_C not
 568 intersecting ϕ' nor ϕ'' is a cycle of Σ , and is therefore orientable. Furthermore, any simple
 569 cycle of Σ_C intersecting ϕ' and/or ϕ'' is homotopic to a cycle separated from both boundaries
 570 of ϕ' and ϕ'' by an open set, and thus is homotopic to a cycle of Σ . It follows that Σ_C is a
 571 closed orientable surface, and thus, thanks to Lemma 5, the genus of Σ_C is $k - 1$.

572 3.1.2 Path-Duplication

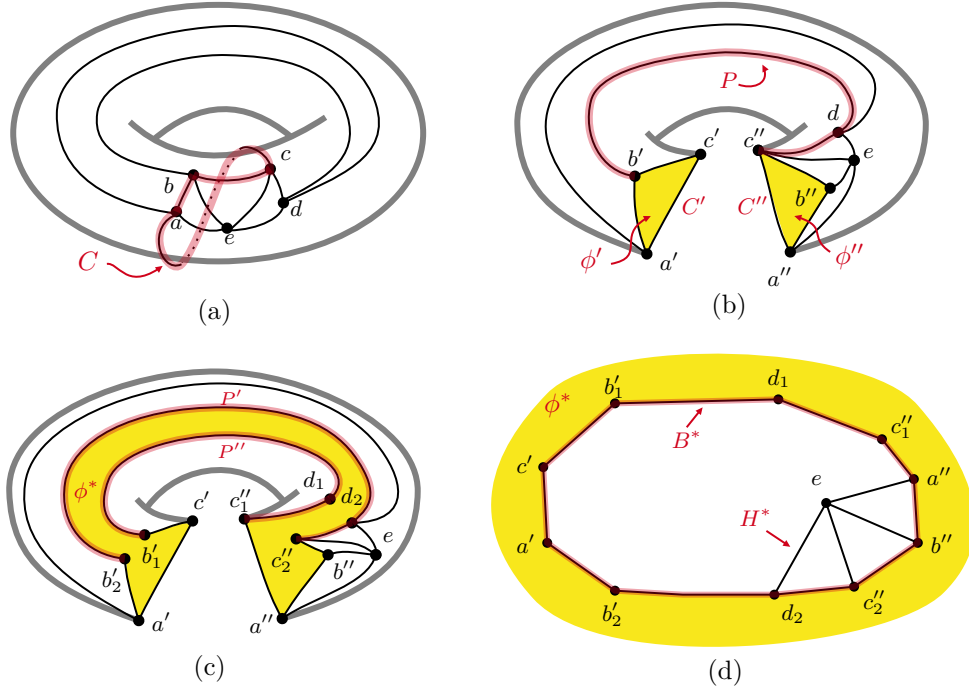
573 Again, let us consider a graph G , an orientable closed surface Σ , and a 2-cell embedding f
 574 of G on Σ . Let χ, ψ be two distinct faces of the embedding, and let $P = (w_0, \dots, w_s)$ be a
 575 simple path (possibly reduced to a single vertex belonging to the two cycles) between χ and
 576 ψ (see Figure 8). That is, P is such that w_0 is on the boundary of ϕ , w_s is on the boundary
 577 of ψ , and no intermediate vertex w_i , $0 < i < s$, is on the boundary of χ or ψ . The path P
 578 enables to define a graph G_P obtained by duplicating the path P in a way similar to the way
 579 the cycle C was duplicated in the previous section. There is only one subtle difference, as the
 580 left and right side of the path cannot be defined at its endpoints. Nevertheless, the left and
 581 right sides of P can still be properly defined all along P , including its extremities, by virtually
 582 “extending” P so that it ends up in the interiors of χ and ψ . Thanks to this path-duplication,
 583 the two faces χ and ψ of G are replaced by a unique face of G_P as illustrated on Figure 8,
 584 reducing the number of faces by one.

585 Remark.

586 Cycle-duplication and path-duplication are typically used conjointly. A basic example, used
 587 for the torus \mathbb{T}_1 in the next section, consists of, first, duplicating a cycle C , then connecting
 588 the two faces resulting from this duplication by a path P , and, finally, duplicating P
 589 for merging these two faces into one single face. Further, for the general case \mathbb{T}_k , $k \geq 1$, k cycles
 590 C_1, \dots, C_k are duplicated, and $2k - 1$ paths P_1, \dots, P_{2k-1} are duplicated for connecting the
 591 $2k$ faces $\phi'_1, \phi''_1, \dots, \phi'_k, \phi''_k$ resulting from the k cycle-duplications, ending up in a unique
 592 face ϕ^* .

593 3.2 Unfolding the Torus

594 As a warm up, we consider the case of a graph embedded on the torus \mathbb{T}_1 , and show how to
 595 “unfold” this embedding.



■ **Figure 9** Unfolding K_5 embedded on the torus \mathbb{T}_1 . The duplication of the non-separating cycle $C = (a, b, c, a)$ creates the faces ϕ' and ϕ'' . Then the duplication of the path $P = (c'', d, b')$ merges ϕ' and ϕ'' into a face ϕ^* . The resulting graph is planar, and ϕ^* can be seen as the infinite face of its embedding.

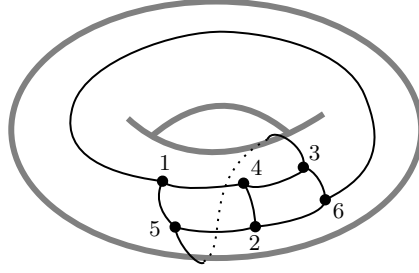
596 **3.2.1 Making a Graph of Genus 1 Planar**

597 Let G be a graph, and let f be a 2-cell embedding of G on $X = \mathbb{T}_1$ — see Figure 9(a) for an
 598 embedding of K_5 on \mathbb{T}_1 , as an illustrative example. Let $C = (v_0, \dots, v_r)$ be a non-separating
 599 orientable cycle of G , e.g., the cycle (a, b, c) on Figure 9(a). Let $C' = (v'_0, \dots, v'_r)$ and $C'' =$
 600 (v''_0, \dots, v''_r) be the two cycles resulting from the duplication of C , e.g., the cycles (a', b', c')
 601 and (a'', b'', c'') on Figure 9(b). The graph G_C with two new faces ϕ' and ϕ'' is connected.
 602 In particular, there exists a simple path $P = (w_0, \dots, w_s)$ in G_C from a vertex $v'_i \in C'$ to a
 603 vertex $v''_j \in C''$, such that every intermediate vertex w_k , $0 < k < s$, is not in $C' \cup C''$, e.g.,
 604 the path (c'', d, b') on Figure 9(b). Note that it may be the case that $i \neq j$. On Figure 9(b),
 605 the path (b'', e, d, b') satisfies $i = j$, but Figure 10 illustrates an embedding of $K_{3,3}$ on \mathbb{T}_1 for
 606 which $i = j$ cannot occur (simply because every vertex of $K_{3,3}$ has degree 3, and thus it has
 607 a single edge not in the cycle). Duplicating P enables to obtain a graph $G_{C,P}$ with a special
 608 face ϕ^* , whose boundary contains all duplicated vertices and only them (see Figure 9(c)).
 609 The details of the vertex-duplications, and of the edge-connections are detailed hereafter.

610 **Connections in path-duplication.**

611 Let $P' = (w'_0, \dots, w'_s)$ and $P'' = (w''_0, \dots, w''_s)$ be the two paths obtained by duplicating P .
 612 In particular, the vertices $w_0 = v'_i$ and $w_s = v''_j$ are both duplicated in w'_0, w''_0 , and w'_s, w''_s ,
 613 respectively. The edges

614 $\{v'_{i-1}, v'_i\}, \{v'_i, v'_{i+1}\}, \{v''_{j-1}, v''_j\},$ and $\{v''_j, v''_{j+1}\}$



■ **Figure 10** $K_{3,3}$ embedded on the torus \mathbb{T}_1 .

615 are replaced by the edges connecting $v'_{i-1}, v'_{i+1}, v''_{j-1}, v''_{j+1}$ to w'_0, w''_0, w'_s, w''_s . For defining
 616 these edges, observe that the path P in \mathbb{T}_1 induces a path $Q = (v_i, w_1, \dots, w_{s-1}, v_j)$ in G
 617 connecting the vertices v_i and v_j of C , such that, in the embedding on \mathbb{T}_1 , the edge $\{v_i, w_1\}$
 618 meets C on one side while the edge $\{w_{s-1}, v_j\}$ meets C on the other side (see Figure 11(a-b)).

619 Figure 11(b) Let us assume, w.l.o.g., that the edges of $C \cup Q$ around v_i are in the order

$$620 \quad \{v_i, v_{i-1}\}, \{v_i, v_{i+1}\}, \{v_i, w_1\}$$

621 when visited counter-clockwise in \mathbb{T}_1 . It follows that the edges of $C \cup Q$ around v_j are in the
 622 order

$$623 \quad \{v_j, v_{j-1}\}, \{v_j, v_{j+1}\}, \{v_j, w_{s-1}\}$$

624 when visited clockwise in \mathbb{T}_1 (see Figure 11(b)). These orders are transferred in G_C , that is,
 625 the edges of $C' \cup P$ around v'_i are in counter-clockwise order

$$626 \quad \{v'_i, v'_{i-1}\}, \{v'_i, v'_{i+1}\}, \{v'_i, w_1\},$$

627 while the edges of $C'' \cup P$ around v''_j are in clockwise order

$$628 \quad \{v''_j, v''_{j-1}\}, \{v''_j, v''_{j+1}\}, \{v''_j, w_{s-1}\},$$

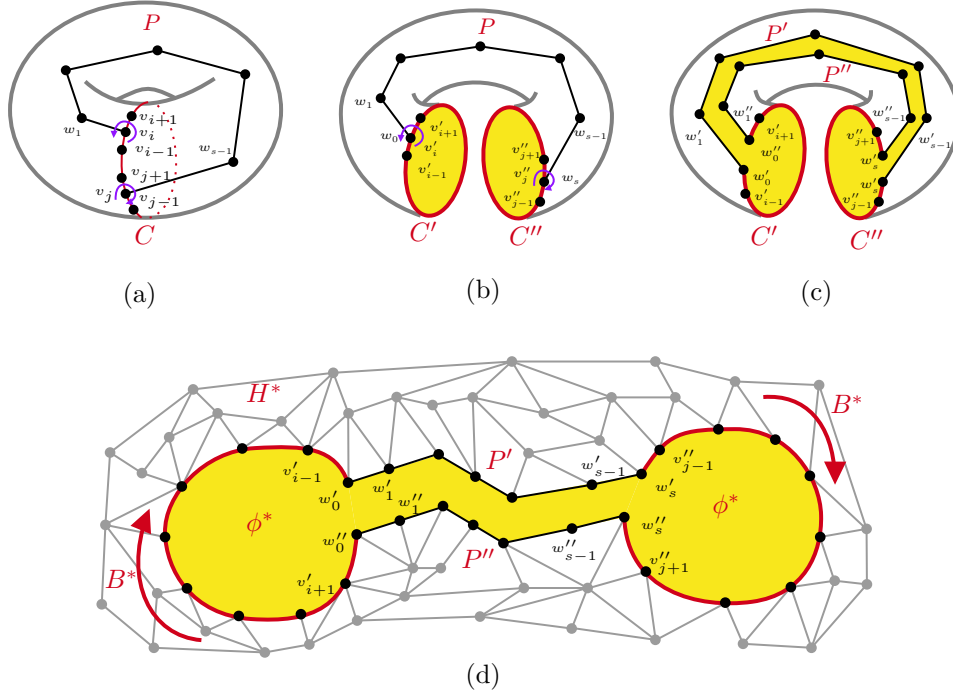
629 as illustrated on Figure 11(b). This guarantees that v'_{i-1} and v''_{j-1} are in the same side of
 630 the path P . More generally, the relative positions of $v'_{i-1}, v'_{i+1}, v''_{j-1}$, and v''_{j+1} w.r.t. P are
 631 as follows. Vertices v'_{i-1} and v''_{j-1} are on the same side of P , while vertices v'_{i+1} and v''_{j+1}
 632 are on the other side of P (see again Figure 11(b)). As a consequence, it can be assumed
 633 that, in the graph $G_{C,P}$ resulting from the duplications of both C and P , the vertices v'_{i-1}
 634 and v''_{j-1} are connected to the end points of P' , while v'_{i+1} and v''_{j+1} are connected to the
 635 end points of P'' . It follows that

$$636 \quad \{w'_0, v'_{i-1}\}, \{w'_s, v''_{j-1}\}, \{w''_0, v'_{i+1}\}, \text{ and } \{w''_s, v''_{j+1}\}$$

637 are edges of $G_{C,P}$ (see Figure 11(c)).

638 **Unfolding.**

639 The embedding f of G on $X = \mathbb{T}_1$ directly induces an embedding of $H^* = G_{C,P}$ on Σ_C , as
 640 illustrated on Figure 11(d). As observed before, the genus of Σ_C is one less than the genus
 641 of Σ . Since $X = \mathbb{T}_1$, it follows that the embedding f of G on \mathbb{T}_1 actually induces a planar



■ **Figure 11** Setting up the connections in path-duplication. A key point, is that v_i is surrounded by the triple (v_{i-1}, v_{i+1}, w_1) counter-clockwise while v_j is surrounded by (v_{j-1}, v_{j+1}, w_s) clockwise. This allows to know where P' and P'' are respectively branched on cycles C' and C'' .

642 embedding f^* of H^* . The faces of this embedding are merely the faces of G , plus another,
 643 special face ϕ^* whose boundary walk is

$$\begin{aligned}
 644 \quad B^* &= (w'_0, w'_1, \dots, w'_s, v'_{j-1}, v'_{j-2}, \dots, v''_0, v''_r, \dots, v''_{j+1}, \\
 645 \quad & \quad \quad \quad w''_s, w''_{s-1}, \dots, w''_0, v'_{i+1}, v'_{i+2}, \dots, v'_r, v'_0, \dots, v'_{i-1}), \\
 646
 \end{aligned}
 \tag{1}$$

647 as displayed on Figure 11(d). For instance, on Figure 9(d), $B^* =$
 648 $(b'_1, d_1, c''_1, a'', b'', c''_2, d_2, b'_2, a', c')$. The face ϕ^* can be pointed out as special, as on
 649 Figure 11(d), or can be made the external face of the embedding of H^* , as on Figure 9(d).
 650 Our interest for H^* , f^* , ϕ^* , and B^* as far as the design of a proof-labeling scheme is
 651 concerned, resides in the fact that, as shown hereafter, they form a (centralized) certificate
 652 for genus 1.

653 3.2.2 Certifying Genus 1

654 Let us first define the notion of *splitting*.

- 655 ► **Definition 8.** A splitting of a graph G into a graph H is a pair $\sigma = (\alpha, \beta)$ of functions,
 656 where $\alpha : V(G) \rightarrow 2^{V(H)}$, and $\beta : E(G) \rightarrow 2^{E(H)}$, such that:
- 657 1. the set $\{\alpha(v) : v \in V(G)\}$ forms a partition of $V(H)$;
 - 658 2. for every $e = \{u, v\} \in E(G)$, $\beta(e)$ is a matching between $\alpha(u)$ and $\alpha(v)$.

659 Note that $\sigma(G)$ may not be connected, even if G is connected. For every $v \in V(G)$, the
 660 vertices $\alpha(v)$ in H are the *avatars* of v in H . The *degree* of a splitting $\sigma = (\alpha, \beta)$ of G into H
 661 is $\max_{v \in V(G)} |\alpha(v)|$, and H is said to be a d -splitting of G whenever $d = \max_{v \in V(G)} |\alpha(v)|$.

662 A vertex $v \in V(G)$ is split in H if $|\alpha(v)| \geq 2$, otherwise it is not split in H . If a vertex v is
 663 not split, we abuse notation by writing $\alpha(v) = v$, i.e., by referring to v as a vertex of G and
 664 as a vertex of H . For any subgraph G' of G , we denote by $\sigma(G')$ the subgraph H' of H with
 665 vertex-set $V(H') = \{\alpha(v) : v \in V(G')\}$, and with edge-set $E(H') = \cup_{e \in E(G')} \beta(e)$. With a
 666 slight abuse of notation, for a splitting $\sigma = (\alpha, \beta)$ of G into H , we often refer to $\sigma(v)$ instead
 667 of $\alpha(v)$ for $v \in V(G)$, and to $\sigma(e)$ instead of $\beta(e)$ for $e \in E(G)$.

668 Let H be a splitting of a graph G for which there exists a 2-splitting U of G such that H
 669 is a 2-splitting of U . Let f be a planar embedding of H , and let ϕ be a face of H embedded
 670 on \mathbb{T}_0 . Let $B = (u_0, \dots, u_N)$ be a boundary walk of ϕ . Let $\sigma_{G,U}$ and $\sigma_{U,H}$ be the splitting of
 671 G into U , and the splitting of U into H , respectively. Let $\sigma_{G,H} = \sigma_{U,H} \circ \sigma_{G,U}$. We say that
 672 (G, H, B, U) is *globally consistent* if there exist vertices $v'_0, \dots, v'_r, v''_0, \dots, v''_r, w'_0, \dots, w'_s,$
 673 w''_0, \dots, w''_s of H such that

$$674 \quad B = (w'_0, \dots, w'_s, v''_{j-1}, \dots, v''_0, v''_r, \dots, v''_{j+1}, w''_s, \dots, w''_0, v'_{i+1}, \dots, v'_r, v'_0, \dots, v'_{i-1})$$

675 where

- 676 ■ for every vertex $u \notin \{v'_k, v''_k : 0 \leq k \leq r\} \cup \{w'_k, w''_k : 0 \leq k \leq s\}$ of H , $\sigma_{G,H}(u) = u$;
- 677 ■ for every $k \in \{1, \dots, s-1\}$, $\sigma_{U,H}^{-1}(\{w'_k, w''_k\}) = w_k \in V(U)$, and $\sigma_{G,U}(w_k) = w_k$;
- 678 ■ for every $k \in \{0, \dots, r\} \setminus \{i, j\}$, $\sigma_{G,U}^{-1}(\{v'_k, v''_k\}) = v_k \in V(U)$, and $\sigma_{U,H}(v_k) = v_k$;
- 679 ■ $\sigma_{U,H}^{-1}(\{w'_0, w''_0\}) = v'_i \in V(U)$, $\sigma_{U,H}^{-1}(\{w'_s, w''_s\}) = v''_j \in V(U)$, $\sigma_{G,U}^{-1}(\{v'_i, v''_i\}) = v_i \in V(G)$,
- 680 and $\sigma_{G,U}^{-1}(\{v'_j, v''_j\}) = v_j \in V(G)$ (note that this applies to both cases $i = j$ and $i \neq j$).

681 **Remark.**

682 The way the vertices of B are listed provides B with a reference direction, say clockwise.
 683 This reference direction is crucial for checking that the two faces of U with respective
 684 boundary walks $v'_i, v'_{i+1}, \dots, v'_r, v'_0, \dots, v'_{i-1}$ and $v''_j, v''_{j-1}, \dots, v''_0, v''_r, \dots, v''_{j+1}$ can be merged
 685 for forming a handle. Global consistency specifies that, for these two faces to be merged,
 686 their directions inherited from the reference direction of B must both be clockwise (cf.,
 687 Figure 11(d)). Indeed, while one face is traversed clockwise with increasing indices, the other
 688 is traversed clockwise with decreasing indices. This matches the specification of handles (cf.
 689 Figure 3).

690 By the construction in Section 3.2.1, for every graph G of genus 1, (G, H^*, B^*, U^*) is
 691 globally consistent, where $H^* = G_{C,P}$, $U^* = G_C$, and B^* is the boundary walk of ϕ^*
 692 displayed in Eq. (1). The following result is specific to the torus, but it illustrates the basis
 693 for the design of our proof-labeling schemes.

694 ► **Lemma 9.** *Let H be a splitting of a graph G , and assume that there exists a planar*
 695 *embedding f of H with a face ϕ and a boundary walk B of ϕ . Let U be a 2-splitting of*
 696 *G such that H is a 2-splitting of U . If (G, H, B, U) is globally consistent, then G can be*
 697 *embedded on the torus \mathbb{T}_1 .*

698 **Proof.** Using the specifications of the splits, the two sub-paths (w'_0, \dots, w'_s) and (w''_0, \dots, w''_s)
 699 of B can be identified by merging each pair of vertices w'_k and w''_k , $k \in \{1, \dots, s-1\}$,
 700 into a single vertex $w_k = \sigma_{U,H}^{-1}(\{w'_k, w''_k\})$ of U , by merging the vertices w'_0 and w''_0 into
 701 a single vertex v'_i of U , and by merging the vertices w'_s and w''_s into a single vertex v''_j
 702 of U . The resulting sequence $v'_i, w_1, \dots, w_{s-1}, v''_j$ forms a path in U connecting two faces
 703 ϕ' and ϕ'' , replacing the face ϕ of the planar embedding f of H , with respective boundary
 704 walks $(v'_0, v'_1, \dots, v'_r)$ and $(v''_r, v''_{r-1}, \dots, v''_0)$, where the vertices are ordered clockwise. These
 705 transformations preserve the planarity of the embedding, that is, U is planar. Next, the two

706 cycles (v'_0, \dots, v'_r) and (v''_0, \dots, v''_r) can be identified, by merging each pair of nodes v'_k and
 707 v''_k into a single node $v_k = \sigma_{G,U}^{-1}(\{v'_k, v''_k\})$ of G . As a result, the two faces ϕ' and ϕ'' are
 708 replaced by a handle, providing an embedding of G on \mathbb{T}_1 . ◀

709 The outcome of Lemma 9 is that (H^*, f^*, ϕ^*, B^*) is essentially a certificate that G can
 710 be embedded on \mathbb{T}_1 (up to also providing the “intermediate” splitting U^* resulting from
 711 cycle-duplication). In the next section, we show how to generalize this construction for
 712 deriving a certificate that a graph G can be embedded on \mathbb{T}_k , $k > 1$.

713 The process described in the previous section for genus 1 can be generalized to larger
 714 genus $k \geq 1$, as follows. Again, let G be a graph, and let f be a 2-cell embedding of G on \mathbb{T}_k .

715 3.2.3 The Face-Duplication Phase

716 Let $\Sigma^{(0)} = \mathbb{T}_k$. As for the torus, let C_1 be a non-separating orientable cycle of $G^{(0)} = G$,
 717 and let us consider the embedding of $G^{(1)} = G_{C_1}^{(0)}$ induced by f , on the surface $\Sigma^{(1)} = \Sigma_{C_1}^{(0)}$
 718 of genus $k - 1$. This operation can be repeated. Indeed, by Lemma 7, there exists a
 719 non separating cycle C_2 of $G^{(1)}$. The graph $G^{(2)} = G_{C_2}^{(1)}$ can be embedded on the surface
 720 $\Sigma^{(2)} = \Sigma_{C_2}^{(1)}$ with one face more than the number of faces of the embedding of $G^{(1)}$ on $\Sigma^{(1)}$,
 721 and thus two more faces than the number of faces of the embedding of G on \mathbb{T}_k . By Lemma 5,
 722 $\Sigma^{(2)}$ has thus genus $k - 2$. See Figure 1(a-b).

723 This process can actually be iterated k times, resulting in a sequence of $k + 1$ graphs
 724 $G^{(0)}, \dots, G^{(k)}$ where $G^{(0)} = G$, and a sequence of $k + 1$ closed surfaces $\Sigma^{(0)}, \dots, \Sigma^{(k)}$ where
 725 $\Sigma^{(0)} = \mathbb{T}_k$. Each graph $G^{(i)}$ is embedded on the closed surface $\Sigma^{(i)}$ of genus $k - i$, as follows.
 726 The embedding of $G^{(0)}$ on $\Sigma^{(0)}$ is the embedding of G on Σ , and, for every $i = 0, \dots, k - 1$, the
 727 embedding of $G^{(i+1)}$ on $\Sigma^{(i+1)}$ is induced by the embedding of $G^{(i)}$ on $\Sigma^{(i)}$, after duplication
 728 of a non-separating cycle C_{i+1} of $G^{(i)}$ into two cycles C'_{i+1} and C''_{i+1} .

729 The closed surface $\Sigma^{(k)}$ is of genus 0, i.e. $\Sigma^{(k)}$ is homeomorphic to the sphere $\mathbb{T}_0 = S^2$
 730 (see Figure 1(b)). The graph $G^{(k)}$ is therefore planar, for it contains k more faces than
 731 the number of faces in G , as two new faces ϕ'_i and ϕ''_i are created at each iteration i , in
 732 replacement to one face ϕ_i , for every $i = 1, \dots, k$.

733 3.2.4 The Face-Reduction Phase

734 The objective is now to replace the $2k$ faces ϕ'_i, ϕ''_i , $i = 0, \dots, k - 1$, by a single face. For this
 735 purpose, let us relabel these faces as ψ_1, \dots, ψ_{2k} (see Figure 1(c)) so that, for $i = 1, \dots, k$,

$$736 \quad \phi'_i = \psi_{2i-1}, \text{ and } \phi''_i = \psi_{2i}.$$

737 Let $\chi_1 = \psi_1$. There exists a simple path P_1 between the two faces χ_1 and ψ_2 . Duplicating
 738 P_1 preserves the fact that the graph $G^{(k+1)} = G_{P_1}^{(k)}$ can be embedded on the sphere \mathbb{T}_0 . By
 739 this duplication, the two faces χ_1 and ψ_2 are merged into a single face χ_2 . Now, there is a
 740 simple path P_2 between the two faces χ_2 and ψ_3 (see Figure 1(d)). Again, duplicating P_2
 741 preserves the fact that the graph $G^{(k+2)} = G_{P_2}^{(k+1)}$ can be embedded on the sphere \mathbb{T}_0 , in
 742 which the two faces χ_2 and ψ_3 are now merged into a single face χ_3 . By iterating this process,
 743 a finite sequence of graphs $G^{(k)}, \dots, G^{(3k-1)}$ is constructed, where, for $i = 0, \dots, 2k - 1$,
 744 the graph $G^{(k+i)}$ is coming with its embedding on \mathbb{T}_0 , and with a set of special faces
 745 $\chi_{i+1}, \psi_{i+2}, \dots, \psi_{2k}$. A path P_{i+1} between χ_{i+1} and ψ_{i+2} is duplicated for merging these
 746 two faces into a single face χ_{i+2} , while preserving the fact that $G^{(k+i+1)} = G_{P_{i+1}}^{(k+i)}$ can be
 747 embedded on the sphere \mathbb{T}_0 .

748 Eventually, the process results in a single face $\phi^* = \chi_{2k}$ of $H^* = G^{(3k-1)}$ (see Figure 1(e)).
 749 This face contains all duplicated vertices. The embedding f of G on \mathbb{T}_k induces a planar
 750 embedding of H^* whose external face is ϕ^* (see Figure 1(f)).

751 3.2.5 Certifying Genus at Most k

752 Conversely, for a graph G of genus k , an embedding of G on \mathbb{T}_k can be induced from the
 753 embedding f^* of H^* on \mathbb{T}_0 , and from the boundary walk B^* of ϕ^* . The latter is indeed
 754 entirely determined by the successive cycle- and path-duplications performed during the
 755 whole process. It contains all duplicated vertices, resulting from the cycles C'_1, \dots, C'_k and
 756 C''_1, \dots, C''_k , and from the paths P'_1, \dots, P'_{2k-1} and P''_1, \dots, P''_{2k-1} . Note that the duplication
 757 process for a vertex may be complex. A vertex may indeed be duplicated once, and then
 758 one of its copies may be duplicated again, and so on, depending on which cycle or path
 759 is duplicated at every step of the process. This phenomenon actually already occurred in
 760 the basic case of the torus \mathbb{T}_1 where the duplications of v_i and v_j were more complex than
 761 those of the other vertices, and were also differing depending on whether $i = j$ or not (see
 762 Section 3.2). Figure 2 illustrates a case in which two cycles C_i and C_j share vertices and
 763 edges in \mathbb{T}_2 , causing a series of duplication more complex than the basic case illustrated on
 764 Figure 1. In particular, a same vertex of H^* may appear several times on the boundary walk
 765 B^* , and a same edge of H^* may be traversed twice, once in each direction.

766 Let H be a splitting of a graph G , let f be a planar embedding of H , and let ϕ be a
 767 face of H embedded on \mathbb{T}_0 . Let $B = (u_0, \dots, u_N)$ be a boundary walk of ϕ , and let \vec{B} be
 768 an arbitrary reference direction given to B , say clockwise. Let $\mathcal{U} = (U_0, \dots, U_{3k-1})$ be a
 769 sequence of graphs such that $U_0 = G$, $U_{3k-1} = H$, and, for every $i \in \{0, \dots, 3k-2\}$, U_{i+1} is
 770 a 2-splitting of U_i . The splitting of U_i into U_{i+1} is denoted by $\sigma_i = (\alpha_i, \beta_i)$. The following
 771 extends the notion of global consistency defined in the case of the torus \mathbb{T}_1 . We say that
 772 $(G, H, \vec{B}, \mathcal{U})$, is *globally consistent* if the following two conditions hold.

- 773 **1. Path-duplication checking.** Let $\chi_{2k} = \phi$, with directed boundary walk $\vec{B}(\chi_{2k}) =$
 774 \vec{B} . For every $i = 0, \dots, 2k-1$, there exist faces $\chi_{i+1}, \psi_{i+2}, \dots, \psi_{2k}^{(i)}$ of U_{k+i} , with
 775 respective directed boundary walks $\vec{B}(\chi_{i+1}), \vec{B}(\psi_{i+2}^{(i)}), \dots, \vec{B}(\psi_{2k}^{(i)})$, and there exist vertices
 776 $u_1^{(i)}, \dots, u_t^{(i)}, v_1^{(i)}, \dots, v_r^{(i)}, w_0^{(i)}, \dots, w_s^{(i)}$, and $w_0''^{(i)}, \dots, w_s''^{(i)}$ of U_{k+i} such that
 - 777 $\vec{B}(\chi_{i+1}) = (w_0^{(i)}, \dots, w_s^{(i)}, v_1^{(i)}, \dots, v_r^{(i)}, w_s''^{(i)}, \dots, w_0''^{(i)}, u_1^{(i)}, \dots, u_t^{(i)})$;
 - 778 $\vec{B}(\psi_{i+1}^{(i)}) = (x, u_1^{(i)}, \dots, u_t^{(i)}, x)$ where $x = \sigma_{k+i-1}^{-1}(\{w_0^{(i)}, w_0''^{(i)}\})$;
 - 779 $\vec{B}(\psi_{i+1}^{(i-1)}) = (y, v_1^{(i)}, \dots, v_r^{(i)}, y)$ where $y = \sigma_{k+i-1}^{-1}(\{w_s^{(i)}, w_s''^{(i)}\})$;
 - 780 $\vec{B}(\psi_j^{(i-1)}) = \vec{B}(\psi_j^{(i)})$.
- 781 **2. Cycle duplication checking.** Let $\phi_1^{(k)} = \chi_1$, and, for $i = 2, \dots, k$, let $\phi_i^{(k)} =$
 782 $\psi_{2i-1}^{(0)}$. For $i = 1, \dots, k$, let $\phi_i''^{(k)} = \psi_{2i}^{(0)}$. For every $i = 1, \dots, k$, there ex-
 783 ists faces $\phi_1^{(i)}, \phi_1''^{(i)}, \dots, \phi_i^{(i)}, \phi_i''^{(i)}$ of U_i with respective directed boundary walks
 784 $\vec{B}(\phi_1^{(i)}), \vec{B}(\phi_1''^{(i)}), \dots, \vec{B}(\phi_i^{(i)}), \vec{B}(\phi_i''^{(i)})$ such that
 - 785 $\vec{B}(\phi_i^{(i)}) = (v_0', v_1', \dots, v_r', v_0')$ and $\vec{B}(\phi_i''^{(i)}) = (v_0'', v_r'', v_{r-1}'', \dots, v_1'', v_0'')$ for some $r \geq 2$,
 - 786 with $|\sigma_{i-1}^{-1}(\{v_j', v_j''\})| = 1$ for every $j = 0, \dots, r$;
 - 787 $\vec{B}(\phi_j^{(i-1)}) = \vec{B}(\phi_j^{(i)})$, and $\vec{B}(\phi_j''^{(i-1)}) = \vec{B}(\phi_j''^{(i)})$.

790 By the construction performed in Sections 3.2.3 and 3.2.4, for every graph G of genus k ,
 791 $(G, H^*, \vec{B}^*, \mathcal{U}^*)$ is globally consistent, where $\mathcal{U}^* = (G^{(0)}, \dots, G^{(3k-1)})$. The following result
 792 generalizes Lemma 9 to graphs of genus larger than 1.

793 ► **Lemma 10.** *Let H be a splitting of a graph G , and assume that there exists a planar*
 794 *embedding f of H with a face ϕ and a boundary walk B of ϕ . Let $\mathcal{U} = (U_0, \dots, U_{3k-1})$ be*
 795 *a series of graphs such that $U_0 = G$, $U_{3k-1} = H$, and, for every $i \in \{0, \dots, 3k-2\}$, U_{i+1}*
 796 *is a 2-splitting of U_i . If $(G, H, \vec{B}, \mathcal{U})$ is globally consistent, then G can be embedded on the*
 797 *torus \mathbb{T}_k .*

798 **Proof.** Condition 1 in the definition of global consistency enables to recover a collection
 799 ψ_1, \dots, ψ_{2k} of faces of U_k . These faces are inductively constructed, starting from the face ϕ
 800 of the planar embedding f of $U_{3k-1} = H$. At each iteration i of the induction, U_{k+i-1} has
 801 faces $\chi_i, \psi_{i+1}^{(i-1)}, \dots, \psi_{2k}^{(i-1)}$ obtained from the faces $\chi_{i+1}, \psi_{i+2}^{(i)}, \dots, \psi_{2k}^{(i)}$ of U_{k+i} by separating
 802 the face χ_{i+1} into two faces χ_i and $\psi_{i+1}^{(i-1)}$ connected by a path, while preserving the other
 803 faces $\psi_{i+2}^{(i)}, \dots, \psi_{2k}^{(i)}$. This operation preserves planarity, and thus, in particular, U_k is planar.

804 The directions of the boundary walks of the faces ψ_1, \dots, ψ_{2k} are inherited from the
 805 original direction given to the boundary walk B . Condition 2 enables to iteratively merge
 806 face ψ_{2i} with face ψ_{2i-1} , $i = 1, \dots, k$, by identifying the vertices of their boundary walks
 807 while respecting the direction of these walks, which guarantees that handles are created (and
 808 not a Klein-bottle-like construction). The process eventually results in the graph U_0 with k
 809 handles, providing an embedding of $U_0 = G$ on \mathbb{T}_k . ◀

810 Thanks to Lemma 10, the overall outcome of this section is that the tuple

$$811 \quad c = (H^*, f^*, \phi^*, B^*, \mathcal{U}^*)$$

812 constructed in Sections 3.2.3 and 3.2.4 is a certificate that G can be embedded on \mathbb{T}_k . This
 813 certificate c and its corresponding verification algorithm are however centralized. In the next
 814 section, we show how to distribute both the certificate c , and the verification protocol.

815 **4 Proof-Labeling Scheme for Bounded Genus Graphs**

816 In this section, we establish our first main result.

817 ► **Theorem 11.** *Let $k \geq 0$, and let \mathcal{G}_k^+ be the class of graphs embeddable on an orientable*
 818 *closed surface of genus at most k . There is a proof-labeling scheme for \mathcal{G}_k^+ using certificates*
 819 *on $O(\log n)$ bits in n -node graphs.*

820 The proof essentially consists of showing how to distribute the centralized certificate

$$821 \quad (H, f, \phi, B, \mathcal{U})$$

822 used in Lemma 10 for a graph G , by storing $O(\log n)$ bits at each vertex of G , while allowing
 823 the vertices to locally verify the correctness of the distributed certificates, that is, in particular,
 824 verifying that (G, H, B, \mathcal{U}) is globally consistent. The rest of the section is entirely dedicated
 825 to the proof of Theorem 11. We start by defining the core of the certificates assigned to the
 826 nodes, called *histories*. Then, we show how to distribute the histories so that every node
 827 stores at most $O(\log n)$ bits, and we describe the additional information to be stored in the
 828 certificates for enabling the liveness and completeness properties of the verification scheme
 829 to hold. Recall that the nodes of G are given arbitrary distinct IDs picked from a set of
 830 polynomial range. The ID of node $v \in V(G)$ is denoted by $\text{id}(v)$. Note that $\text{id}(v)$ can be
 831 stored on $O(\log n)$ bits.

4.1 Histories

The description of the certificates is for positive instances, that is, for graphs $G \in \mathcal{G}_k^+$. For such an instance G , the prover performs the construction of Section 3.2.2, resulting in the series of 2-splitting graphs $G^{(0)} = G, G^{(1)}, \dots, G^{(2k-2)}, G^{(2k-2)} = H^*$, a planar embedding f of H^* , and the identification of a special face ϕ^* in this embedding, with boundary walk B^* . The successive duplications experienced by a vertex v of the actual graph G during the face-duplication and face-reduction phases resulting in H^* can be encoded as a rooted binary tree unfolding these duplications, called *history*.

For every vertex v of G , the history of v is denoted by $\mathbf{h}(v)$. The history of v is a rooted binary tree of depth $3k - 1$ (all leaves are at distance $3k - 1$ from the root). For $\ell = 0, \dots, 3k - 1$, the level ℓ of $\mathbf{h}(v)$ consists of the at most 2^ℓ nodes at distance ℓ from the root. The internal nodes of $\mathbf{h}(v)$ with two children are called *binary* nodes, and the internal nodes with one child are called *unary*.

- For $\ell = 0, \dots, k - 1$, the edges connecting nodes of level ℓ to nodes of level $\ell + 1$ are corresponding to the duplication of the cycle $C_{\ell+1}$ in $G^{(\ell)}$ (cf. Section 3.2.3), and,
- for $\ell = 0, \dots, 2k - 1$, the edges connecting nodes of level $k + \ell$ to nodes of level $k + \ell + 1$ are corresponding to the duplication of the path $P_{\ell+1}$ in $G^{(k+\ell)}$ (cf. Section 3.2.4).

The nodes of $\mathbf{h}(v)$ are provided with additional information, as follows.

4.1.1 Vertices and Adjacencies in the Splitting Graphs

For every $\ell = 1, \dots, 3k - 1$, every node x at level ℓ in $\mathbf{h}(v)$ is provided with the vertex u of $G^{(\ell)}$ it corresponds to, after the duplications of v corresponding to the path from the root to x . In particular, each leaf of $\mathbf{h}(v)$ is provided with the single vertex of $H^* = G^{(3k-1)}$ it corresponds to. Specifically, each internal node x of $\mathbf{h}(v)$ is provided with the set S_x of vertices of H^* marked at the leaves of the subtree of $\mathbf{h}(v)$ rooted at x . For a leaf x , $S_x = \{u\}$, where u is the avatar of v in H^* corresponding to the path from the root to the leaf x . Note that, for two distinct nodes at level ℓ in $\mathbf{h}(v)$, we have $S_x \cap S_y = \emptyset$.

The $3k - 1$ splittings successively performed starting from G are 2-splittings, from which it follows that every vertex of G is split a constant number of times for a fixed k . The $\nu \geq 1$ avatars of $v \in V(G)$ in H^* are labeled $(\text{id}(v), 1), \dots, (\text{id}(v), \nu)$. It follows that the ν leaves of $\mathbf{h}(v)$ are respectively labeled $(\text{id}(v), 1), \dots, (\text{id}(v), \nu)$. For every node x of $\mathbf{h}(v)$, each set S_x is a subset of $\{(\text{id}(v), 1), \dots, (\text{id}(v), \nu)\}$, and thus these sets S_x can be stored on $O(\log n)$ bits.

Every node x of $\mathbf{h}(v)$ at level $\ell \in \{0, \dots, 3k - 1\}$, which, as explained above, corresponds to a vertex of $G^{(\ell)}$, is also provided with the set N_x of the neighbors of S_x in $G^{(\ell)}$. The set N_x has the form $N_x = \{X_1, \dots, X_d\}$ for some $d \geq 1$, where, for $i = 1, \dots, d$, X_i is a vertex of $G^{(\ell)}$ corresponding to a set of avatars in H^* of some neighbor w of v in G .

Since some vertices $v \in V(G)$ may have arbitrarily large degree (up to $n - 1$), the sets N_x may not be storable using $O(\log n)$ bits. As a consequence, some histories may not be on $O(\log n)$ bits, and may actually be much bigger. Nevertheless, a simple trick using the fact that graphs with bounded genus have bounded degeneracy (cf. Lemma 6) allows us to reassign locally the set N_x in the histories so that every node of G stores $O(\log n)$ bits only.

4.1.2 Footprints

Every node x of $\mathbf{h}(v)$ at level $\ell \in \{0, \dots, 3k - 1\}$ is provided with a (possibly empty) set F_x of ordered triples of the form (X, Y, Z) where $X \in N_x$, $Y = S_x$, and $Z \in N_x$, called *footprints*. Intuitively, each footprint encodes edges $\{X, Y\}$ and $\{Y, Z\}$ of $G^{(\ell)}$ occurring in:

- a boundary walk of one of the faces ϕ'_i or ϕ''_i , $i = 1, \dots, \ell$, if $\ell \leq k$, or

877 ■ a boundary walk of one of the faces $\chi_{\ell-k}, \psi_{\ell-k+1}, \dots, \psi_{2k}$, otherwise.

878 Note that these two edges are actually directed, from X to Y , and from Y to Z , reflecting
879 that the boundary walk is traveled in a specific direction, inherited from some a priori
880 direction, say clockwise, given to the boundary walk B^* of the face $\phi^* = \chi_{2k}$ (hence the
881 terminology “footprints”).

882 Note that a same vertex of $G^{(\ell)}$ may appear several times in the boundary walk of a face,
883 and a same edge may appear twice, once in every direction. Therefore, a same node x of $h(v)$
884 may be provided with several footprints, whose collection form the set F_x , which may be of
885 non-constant size. On the other hand, for a fixed k , a constant number of boundary walks
886 are under concern in total, from which it follows that even if a node x at level ℓ of $h(v)$ must
887 store a non-constant number of footprints in F_x , each of x 's incident edges in $G^{(\ell)}$ appears
888 in at most two footprints of F_x . We use this fact, together with the bounded degeneracy of
889 the graphs of bounded genus, for reassigning locally the sets F_x in the histories so that every
890 node of G stores $O(\log n)$ bits only.

891 4.1.3 Types

892 Last, but not least, for every node x of $h(v)$, each of the two (directed) edges (X, Y) and
893 (Y, Z) in every footprint (X, Y, Z) in F_x also comes with a *type* in

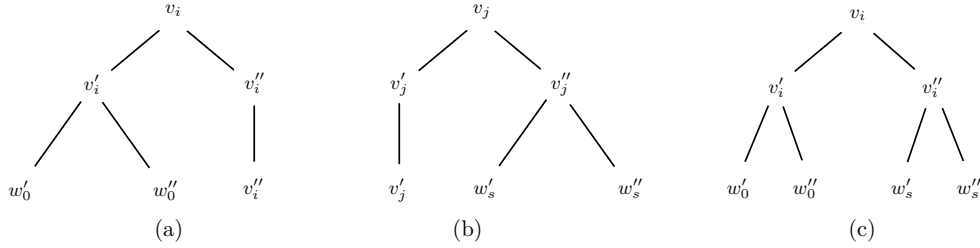
$$894 \quad \mathsf{T}_k = \{C'_1, \dots, C'_k, C''_1, \dots, C''_k, P'_1, \dots, P'_{2k-1}, P''_1, \dots, P''_{2k-1}\},$$

895 which reflects when this edge was created during the cycle- and path-duplications.

896 Example.

897 Figure 12 provides examples of histories for some vertices of G in the case displayed on
898 Figure 11. Figure 12(a-b) display the histories of v_i and v_j whenever $i \neq j$, while Figure 12(c)
899 displays the histories of $v_i = v_j$ whenever $i = j$. In this latter case, the leaves w'_0, w''_0, w'_s, w''_s
900 may be labeled as $(\text{id}(v), 1), (\text{id}(v), 2), (\text{id}(v), 3), (\text{id}(v), 4)$, respectively. Then v'_i is labeled
901 $S_{v'_i} = \{(\text{id}(v), 1), (\text{id}(v), 2)\}$, while v''_i is labeled $S_{v''_i} = \{(\text{id}(v), 3), (\text{id}(v), 4)\}$, and root is
902 labeled $S_{v_i} = \{(\text{id}(v), 1), (\text{id}(v), 2), (\text{id}(v), 3), (\text{id}(v), 4)\}$. The neighborhoods N_x of these
903 nodes x of $h(v_i)$ are depending on the graphs $G^{(0)} = G, G^{(1)}$, and $G^{(2)} = H^*$. Assuming that
904 B^* is directed clockwise, as displayed on Figure 11(d), the leaf w'_0 is provided with footprint
905 (v'_{i-1}, w'_0, w'_1) while the leaf w''_0 is provided with footprint (w''_1, w''_0, v'_{i+1}) . Similarly, w'_s and
906 w''_s are respectively provided with footprint $(w'_{s-1}, w'_s, v''_{i-1})$ and $(v''_{i+1}, w''_s, w''_{s-1})$, where the
907 various nodes in these footprints are encoded depending on their labels in H^* , which depend
908 on the IDs given to the neighbors of v_i in G . The footprint at v'_i is $(v'_{i-1}, v'_i, v'_{i+1})$, while
909 the footprint at v''_i is $(v''_{i+1}, v''_i, v''_{i-1})$. In both case, the directions of the edges are inherited
910 from the initial clockwise direction of the boundary walk B^* . The directed edges (v'_{i-1}, v'_i)
911 and (v'_i, v'_{i+1}) receives type C'_1 , while the directed edges (v''_{i+1}, v''_i) and (v''_i, v''_{i-1}) receives
912 type C''_2 . The four edges $(v'_{i-1}, w'_0), (w''_0, v'_{i+1}), (v''_{i+1}, w''_s)$, and (w'_s, v''_{i-1}) are respectively
913 inheriting the types C'_1, C'_1, C''_1 , and C''_1 of the four edges $(v'_{i-1}, v'_i), (v'_i, v'_{i+1}), (v''_{i+1}, v''_i)$, and
914 (v''_i, v''_{i-1}) . The directed edges (w'_0, w'_1) and (w'_{s-1}, w'_s) receive type P'_1 , while the directed
915 edges (w''_1, w''_0) and (w''_s, w''_{s-1}) receive type P''_1 . Observe that the footprints are constructed
916 upward the histories, while the types are assigned downward those trees.

917 We now detail how the footprints are constructed in general, and how the types are
918 assigned to the edges of the footprints.



■ **Figure 12** Examples of histories.

919 4.1.4 Construction of the Footprints

920 Let us give an arbitrary orientation, say clockwise, to the boundary walk B^* of the special
 921 face ϕ^* of H^* . This orientation induces footprints $(\text{pred}(u), u, \text{succ}(u)) \in F_x$ given to every
 922 leaf x of every history $h(v)$, $v \in V(G)$. The vertex $\text{pred}(u) \in V(H^*)$ is the predecessor
 923 of the avatar $u \in V(H^*)$ of v in H^* , and $\text{succ}(u) \in V(H^*)$ is its successor. Note that
 924 some leaves x have $F_x = \emptyset$, whenever the corresponding node u in H^* does not belong to
 925 the boundary walk B^* . On the other hand, as a same node can be visited several times
 926 when traveling along the boundary walk B^* , some leaves may be given several footprints
 927 $(\text{pred}_1(u), u, \text{succ}_1(u)), \dots, (\text{pred}_d(u), u, \text{succ}_d(u))$ in F_x , for some $d \geq 1$. The footprints
 928 provided to the internal nodes of the histories of the vertices of G are given in a way
 929 consistent with the orientation of B^* . More specifically, the footprints are constructed
 930 upward the histories, as follows.

931 Hereafter, the symbol “ $\xrightarrow{\ell}$ ” stands for the operation performed when going from level
 932 $\ell - 1$ to level ℓ , or vice-versa, from level ℓ to level $\ell - 1$. For instance, for three sets S, S', S''
 933 of vertices from H^* , the relation

$$934 \quad S \xrightarrow{\ell} S', S''$$

935 states that the vertices S' and S'' of $G^{(\ell)}$ are the results of a cycle- or path-duplication
 936 experienced by the vertex S occurring from $G^{(\ell-1)}$ to $G^{(\ell)}$, i.e., the vertex $S = S' \cup S''$ of
 937 $G^{(\ell-1)}$ is split into two avatars, S' and S'' , in $G^{(\ell)}$. If $\ell \leq k$, the split was caused by a
 938 cycle-duplication, otherwise it was caused by a path-duplication. Similarly, for two footprints
 939 F' and F'' at two nodes at level ℓ , children of a same binary node, the relation

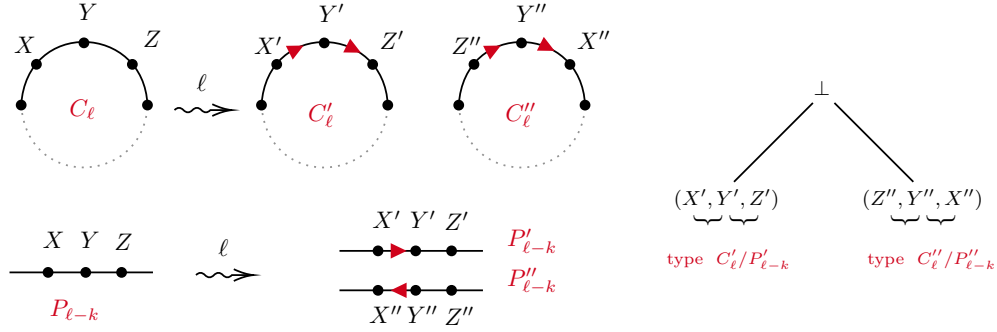
$$940 \quad F', F'' \xrightarrow{\ell} F$$

941 states that, when going upward a history, the two footprints F' and F'' of level ℓ generate
 942 the footprint F at level $\ell - 1$.

943 Three rules, called *Elementary*, *Extremity*, and *Vacancy*, are applied for the construction
 944 of the footprints. Their role is to “role back” the boundary walk B^* of the special face ϕ^* in
 945 the planar embedding of H^* . Each edge of the boundary walk B^* is indeed resulting from
 946 some duplication, of either a cycle or a path. The footprints encode the histories of all edges
 947 of the boundary walk B^* in all graphs $G^{(\ell)}$, $0 \leq \ell \leq 3k - 1$, including when the edges were
 948 created (referred to as the *types* of the edges), and what were their successive extremities
 949 when those extremities are duplicated.

950 **Elementary rule.** Assuming $X \xrightarrow{\ell} X', X''$, $Y \xrightarrow{\ell} Y', Y''$, and $Z \xrightarrow{\ell} Z', Z''$, the elementary
 951 rule matches two footprints of two children Y' and Y'' , and produces none at the parent Y :

$$952 \quad (X', Y', Z'), (Z'', Y'', X'') \xrightarrow{\ell} \perp.$$



■ **Figure 13** Footprint construction, and type assignment: Elementary rule.

953 The Elementary rule applies to the case of cycle duplication, as well as to the case of
 954 path-duplication, but to the internal nodes of the path only (see Figure 13). When two
 955 cycles are merged (as the opposite to cycle duplication), their faces are glued together, and
 956 disappear. Similarly, when two paths are merged (as the opposite of path-duplication),
 957 the resulting path is of no use, and it can be discarded. Note that the two footprints
 958 (X', Y', Z') and (Z'', Y'', X'') are ordered in opposite directions. This matches the
 959 requirement for correctly glueing the borders of two faces in order to produce a handle
 960 (see Figures 3 and 7). This also matches the way the two copies of a path P_i are traversed
 961 when traveling along the boundary walk B^* in clockwise direction (cf. Eq. (1) and
 962 Figure 8).

963 **Extremity rule.** This rule applies only for levels $\ell > k$. It has two variants, defined below.

964 *Single extremity rule.* Assuming $X' \xrightarrow{\ell} X'$, $X'' \xrightarrow{\ell} X''$, $Y \xrightarrow{\ell} Y, Y''$, and $Z \xrightarrow{\ell} Z', Z''$,
 965 the single extremity rule matches two footprints of two children Y' and Y'' , and
 966 produces one footprint at the parent Y :

$$967 \quad (X', Y', Z'), (Z'', Y'', X'') \xrightarrow{\ell} (X', Y, X'').$$

968 *Double extremity rule.* Assuming $X' \xrightarrow{\ell} X'$, $X'' \xrightarrow{\ell} X''$, $Y \xrightarrow{\ell} Y', Y''$, $Z' \xrightarrow{\ell} Z'$, and
 969 $Z'' \xrightarrow{\ell} Z''$, the double extremity rule matches two footprints of two children Y' and
 970 Y'' , and produces two footprints at the parent Y :

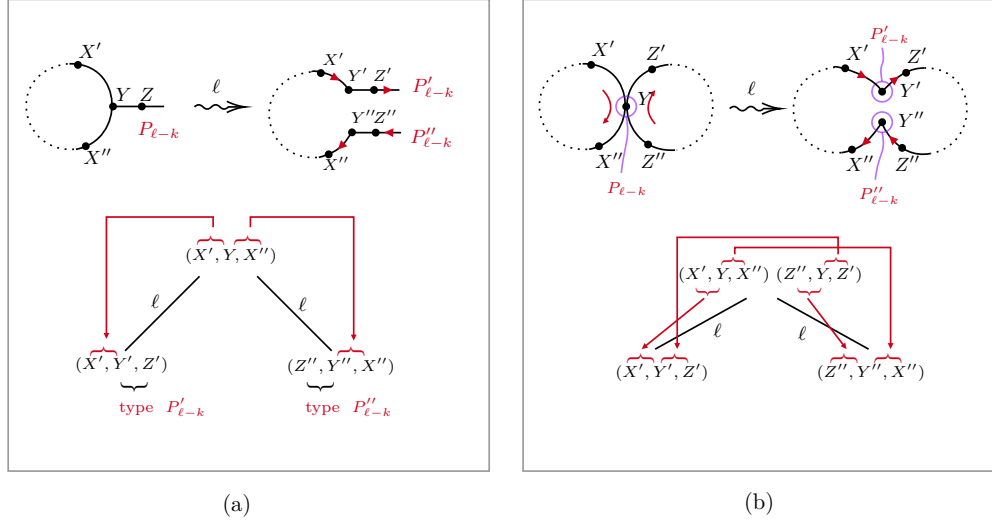
$$971 \quad (X', Y', Z'), (Z'', Y'', X'') \xrightarrow{\ell} \{(X', Y, X''), (Z'', Y, Z')\}.$$

972 The Extremity rule refers to path duplication only (i.e., to levels $\ell > k$), as displayed
 973 on Figure 14. It is dedicated to the extremities of the path considered at this phase
 974 (see Figure 8). The Single extremity rule (cf. Figure 14(a)) handles the standard case
 975 in which the path is not trivial (i.e., reduced to a single vertex), whereas the Double
 976 extremity rule (cf. Figure 14(b)) handles the case in which the path connecting two
 977 faces is reduced to a single vertex Y (i.e., the two corresponding cycles share at least
 978 one vertex Y). Then only the vertex Y is split during the path duplication, while its
 979 four neighbors X', X'', Z' , and Z'' remain intact.

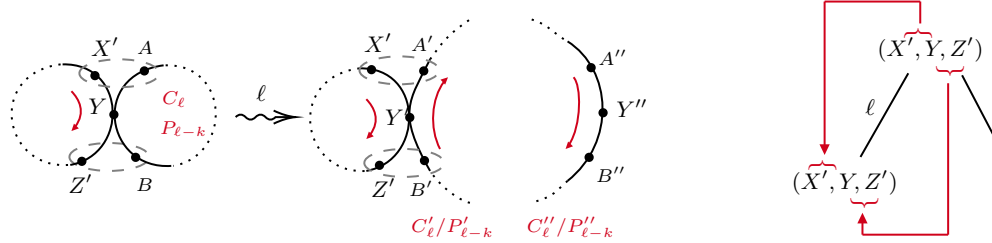
980 **Vacancy rule.** The vacancy rule simply forwards a footprint upward:

$$981 \quad (X', Y, Z') \xrightarrow{\ell} (X, Y, Z)$$

982 with $X \xrightarrow{\ell} X', X''$ (resp., $Y \xrightarrow{\ell} Y', Y''$, and $Z \xrightarrow{\ell} Z', Z''$), unless $X \xrightarrow{\ell} X$ (resp., $Y \xrightarrow{\ell} Y$,
 983 and $Z \xrightarrow{\ell} Z$), in which case $X = X'$ (resp., $Y = Y'$, and $Z = Z'$).



■ **Figure 14** Footprint construction, and type assignment: Extremity rule.



■ **Figure 15** Footprint construction, and type assignment: Vacancy rule.

984 The Vacancy rule handles the case where one of the twin nodes carries a footprint
 985 (X', Y', Z') (resp., (X'', Y'', Z'')), which is copied to the parent node, after updating the
 986 vertices in case the latter experienced duplications (see Figure 15).

987 4.1.5 Assigning Types to Footprints

988 The types in \mathbb{T}_k are assigned to the edges of the footprints, downwards the histories, as
 989 follows.

- 990 ■ If the footprints (X', Y', Z') and (Z'', Y'', X'') are matched by application of the Elementa-
 991 rary rule at level ℓ , then the two (directed) edges (X', Y') and (Y', Z') (resp., (Z'', Y'')
 992 and (Y'', X'')) of $G^{(\ell)}$ are given type C'_{ℓ} (resp., C''_{ℓ}) if $\ell \leq k$, and $P'_{\ell-k}$ (resp. $P''_{\ell-k}$)
 993 otherwise. See Figure 13.
- 994 ■ If the footprints (X', Y', Z') and (Z'', Y'', X'') are matched by application of the Single
 995 extremity rule at level ℓ , then the two edges (X', Y) and (Y, X'') adopt the types of the
 996 edges (X', Y') and (Y'', X'') , respectively, while the two edges (Y', Z') and (Z'', Y'') are
 997 given type $P'_{k-\ell}$ and $P''_{k-\ell}$, respectively. See Figure 14(a).
- 998 ■ If the footprints (X', Y', Z') and (Z'', Y'', X'') are matched by application of the Double
 999 extremity rule at level ℓ , then the four edges (X', Y') , (Y', Z') , (Z'', Y'') , and (Y'', X'')
 1000 adopt the types of the edges (X', Y) , (Y, Z') , (Z'', Y) , and (Y, X'') , respectively. See
 1001 Figure 14(b)
- 1002 ■ If the footprint (X', Y', Z') is forwarded upward as (X, Y, Z) by application of the

1003 Vacancy rule, then (X', Y') , and (Y', Z') adopt the types of the edges (X, Y) , and (Y, Z) ,
 1004 respectively. See Figure 15.

1005 We have now all the ingredients to state what will be proved as sufficient to certify that
 1006 a graph G has genus at most k .

1007 4.2 Assignment of the Histories to the Certificates

1008 As it was mentioned in Section 4.1, the history $h(v)$ of a node v of the actual graph G may
 1009 not be on $O(\log n)$ bits. The reason for that is that, even if G has a bounded genus k , the
 1010 node v may have an arbitrarily large degree. As a consequence, the sum of the degrees of
 1011 its v 's avatars in each of the graphs $G^{(0)}, \dots, G^{(3k-1)}$ may be arbitrarily large. This has
 1012 direct consequences not only on the memory requirement for storing the neighborhood N_x
 1013 of each node $x \in h(v)$, but also on the number of footprints to be stored in F_x . In both
 1014 cases, this memory requirement may exceed $O(\log n)$ bits. On the other hand, every graph
 1015 G of bounded genus is sparse, which implies that the average degree of G , and of all its
 1016 splitting graphs $G^{(0)}, \dots, G^{(3k-1)}$ is constant. Therefore, the average memory requirement
 1017 per vertex v for storing all the histories $h(v)$, $v \in V(G)$, is constant. Yet, it remains that
 1018 some vertices $v \in V(G)$ may have large histories, exceeding $O(\log n)$ bits.

1019 The simple trick under this circumstances (cf., e.g., [21]) is to consider the space-complexity
 1020 of the histories not per node of G , but per edge. Indeed, the space-complexity of the
 1021 information related to each edge e of G , as stored in the histories, is constant, for *every*
 1022 edge e . For instance, at a node x of level ℓ in some historie $h(v)$, instead of storing N_x at v ,
 1023 one could virtually store every edge $\{S_x, S_y\}$, $S_y \in N_x$, on the edge $\{v, w\}$ of G , where w is
 1024 the neighbor of v in G with avatar S_y in $G^{(\ell)}$.

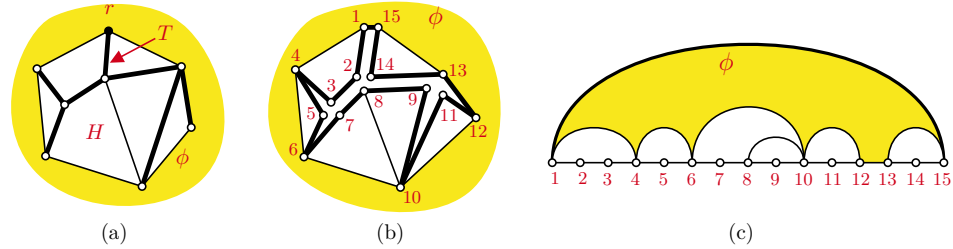
1025 Let us define a *line* proof-labeling scheme as a proof-labeling scheme in which certificates
 1026 are not only assigned to the vertices of G , but also to the edges of G (i.e., to vertices of the
 1027 line-graph of G). In a line proof-labeling scheme, the vertices forge their decisions not only
 1028 on their certificates and on the certificates assigned to their adjacent vertices, but also on the
 1029 certificates assigned to their incident edges. Our interest for the concept of line proof-labeling
 1030 scheme is expressed in the following result, after having recalled that, thanks to Lemma 6,
 1031 every graph of genus at most k is d -degenerate for some constant d depending on k .

1032 **► Lemma 12.** *Let $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $f(n) \in \Omega(\log(n))$. Let $d \geq 1$, and let \mathcal{G} be a graph
 1033 family such that every graph in \mathcal{G} is d -degenerate. If \mathcal{G} has a line proof-labeling scheme with
 1034 certificate size $O(f(n))$ bits, then \mathcal{G} has a proof-labeling scheme with certificate size $O(f(n))$
 1035 bits.*

1036 **Proof.** Let (\mathbf{p}, \mathbf{v}) be line proof-labeling scheme for \mathcal{G} . For $G \in \mathcal{G}$, the prover \mathbf{p} assigns
 1037 certificate $\mathbf{p}(v)$ to every node $v \in V(G)$, and certificate $\mathbf{p}(e)$ to every edge $e \in E(G)$. Since
 1038 G is d -degenerate, there exists a node v of G with degree $d_v \leq d$. Let $c(v)$ be the certificate
 1039 of v defined as

$$1040 \quad c(v) = \left(\mathbf{p}(v), \{(\text{id}(u_1), \mathbf{p}(e_1)), \dots, (\text{id}(u_{d_v}), \mathbf{p}(e_{d_v}))\} \right),$$

1041 where u_1, \dots, u_{d_v} are the d_v neighbors of v in G , and, for every $i = 1, \dots, d_v$, $e_i = \{v, u_i\}$.
 1042 Since the IDs can be stored on $O(\log n)$ bits, and since $f(n) \in \Omega(\log n)$, we get that $c(v)$ can
 1043 be stored on $O(f(n))$ bits. This construction can then be repeated on the graph $G' = G - v$,
 1044 which still has degeneracy at most d . By iterating this construction, all nodes are exhausted,
 1045 and assigned certificates on $O(f(n))$ bits, containing all the information originally contained



■ **Figure 16** Illustration of the PLS for planarity in [21].

1046 in the node- and edge-certificates assigned by \mathbf{p} . We complete the proof by observing that,
 1047 for every edge $e = \{u, v\}$ of G , the certificate $\mathbf{p}(e)$ assigned by \mathbf{p} to e can be found either in
 1048 $c(u)$ or in $c(v)$. This suffices for simulating the behavior of \mathbf{v} , and thus for the design of a
 1049 standard proof-labeling scheme for \mathcal{G} . ◀

1050 4.3 Certifying Planarity

1051 In this section, we show how to certify that H is a planar embedding with a special face ϕ
 1052 with boundary walk B . For this purpose, we just need to slightly adapt a recent proof-labeling
 1053 scheme for planarity [21].

1054 ► **Lemma 13.** *There exists a proof-labeling scheme for certifying that a given graph H has a*
 1055 *planar embedding f , including a face ϕ with boundary walk B .*

1056 **Proof.** Let H be a planar graph with a planar embedding f . The scheme for planarity
 1057 in [21] constructs the certificates as follows (cf. Figure 16). Let T be an arbitrary spanning
 1058 tree of H , and let us root T at a vertex $r \in V(H)$ on the outer face ϕ , as displayed on
 1059 Figure 16(a). The tree T is “flattened” into a cycle C in a splitting H' of H by replacing
 1060 every vertex $v \in V(H)$ by as many vertices as the number of times v is visited by a DFS
 1061 traversal of T starting from r (see Figure 16(b)). The scheme in [21] certifies the cycle C ,
 1062 viewed as a path P whose two extremities are avatars of r , with respective DFS numbers 1
 1063 and $2n - 1$, plus an edge connecting these two avatars (see Figure 16(c)). A property of
 1064 this construction taken from [21] is that the vertices of H on the outer face ϕ are those
 1065 which have at least one avatar in H' such that no co-tree edges “jumps over it” when the
 1066 vertices are displayed as on Figure 16(c). For instance, the avatars 1, 4, 6, 10, 12, 13, 15 have
 1067 no co-tree edges jumping over them, and indeed these avatars are the ones of the vertices on
 1068 the boundary of the outer face ϕ . The scheme of [21] is precisely based on a local encoding
 1069 of the “lower edge” jumping over every avatars in H' . It follows that this scheme suffices for
 1070 certifying not only the planarity of H , but also that ϕ is a face of H with boundary B . ◀

1071 4.4 Local Consistency

1072 Let H be a splitting of a graph G , let f be a planar embedding of H , and let ϕ be a face of
 1073 H with boundary walk B directed, say, clockwise. The directed boundary walk B is denoted
 1074 by \vec{B} . Let $\mathbf{h}(G) = \{\mathbf{h}(v), v \in V(G)\}$ be a collection of histories for the vertices of G , of depth
 1075 $3k - 1$, for some $k \geq 1$. We say that $(G, H, \vec{B}, \mathbf{h}(G))$ is *locally consistent* if the following
 1076 holds.

- 1077 1. There exists a sequence of graphs U_0, \dots, U_{3k-1} with $U_0 = G$, $U_{3k-1} = H$, and, for every
 1078 $0 \leq \ell < 3k - 1$, $U_{\ell+1}$ is a degree-2 splitting of U_ℓ , such that, for every $v \in V(G)$, and for
 1079 every $\ell = 0, \dots, 3k - 1$, every node x at level ℓ of $\mathbf{h}(v)$ satisfies that S_x is a vertex of U_ℓ ,
 1080 the neighborhood of S_x defined in N_x is consistent with the neighborhood of S_x in U_ℓ ,
 1081 and the footprints in F_x contains edges of U_ℓ . Moreover, if x has two children x' and x''
 1082 in $\mathbf{h}(v)$, then there are exactly two footprints, one in $E_{x'}$ and one in $E_{x''}$, for which the
 1083 Elementary rule or the Extremity rule was applied, all the other footprints in $E_{x'}$ and
 1084 $E_{x''}$ being subject to the Vacancy rule. Furthermore, if x has a unique child x' , then all
 1085 footprints in $E_{x'}$ are subject to the Vacancy rule. Finally, the typing is consistent with
 1086 the specified typing rules.
- 1087 2. The collection of footprints at the leaves of the histories in $\mathbf{h}(G)$ can be ordered as
 1088 $(x_0, y_0, z_0), \dots, (x_N, y_N, z_N)$ such that, $y_i = z_{i-1} = x_{i+1}$ for every $i = 0, \dots, N$, and
 1089 $\vec{B} = (y_0, \dots, y_N)$.
- 1090 3. For every $\ell = 1, \dots, 2k - 1$, the following must be satisfied:
- 1091 a. the collection of footprints at the nodes at level $k + \ell$ whose both edges
 1092 have type P'_ℓ (resp., type P''_ℓ) in the histories in $\mathbf{h}(G)$ can be ordered as
 1093 $(X'_0, Y'_0, Z'_0), \dots, (X'_{s_\ell}, Y'_{s_\ell}, Z'_{s_\ell})$ (resp., $(Z''_0, Y''_0, X''_0), \dots, (Z''_{s_\ell}, Y''_{s_\ell}, X''_{s_\ell})$), for some $s_\ell \geq$
 1094 0 , such that:
- 1095 i. for every $i = 0, \dots, s_\ell$, $Y_i \xrightarrow{k+\ell} \{Y'_i, Y''_i\}$;
 1096 ii. for every $i = 1, \dots, s_\ell$, $Y'_i = Z'_{i-1}$ and $Y''_i = Z''_{i-1}$;
 1097 iii. for every $i = 0, \dots, s_\ell - 1$, $Y'_i = X'_{i+1}$ and $Y''_i = X''_{i+1}$;
- 1098 b. the collection of footprints at the nodes at level $k + \ell$ whose both edges have type
 1099 $C'_{\lceil \frac{\ell+1}{2} \rceil}$ if $\ell + 1$ is odd, or type $C''_{\frac{\ell+1}{2}}$ if $\ell + 1$ is even, can be ordered as $(X_0, Y_0, Z_0), \dots,$
 1100 $(X_{r_\ell}, Y_{r_\ell}, Z_{r_\ell})$, for some $r_\ell \geq 0$ such that, for every $i = 1, \dots, r_\ell$, $Y_i = Z_{i-1} = X_{i+1}$;
- 1101 c. the collection of footprints at the nodes at level $k + \ell$ whose both edges have same
 1102 type $P'_1, P''_1, \dots, P'_{\ell-1}, P''_{\ell-1}, C'_1, C''_1, \dots, C'_{\lceil \ell/2 \rceil}, C''_{\lceil \ell/2 \rceil}$, or $C'_{(\ell+1)/2}$ if $\ell + 1$ is even, can
 1103 be ordered as $(X_0, Y_0, Z_0), \dots, (X_{t_\ell}, Y_{t_\ell}, Z_{t_\ell})$, for some $t_\ell \geq 0$, such that for every
 1104 $i = 1, \dots, t_\ell$, $Y'_i = Z'_{i-1}$ and $Y''_i = Z''_{i-1}$;
- 1105 4. For every $\ell = 1, \dots, k$, the collection of footprints at the nodes at level ℓ whose both edges
 1106 have type C'_ℓ (resp., type C''_ℓ) in the histories in $\mathbf{h}(G)$ can be ordered as $(X'_0, Y'_0, Z'_0), \dots,$
 1107 $(X'_{r_\ell}, Y'_{r_\ell}, Z'_{r_\ell})$ (resp., $(Z''_0, Y''_0, X''_0), \dots, (Z''_{r_\ell}, Y''_{r_\ell}, X''_{r_\ell})$), for some $r_\ell \geq 0$, such that:
- 1108 a. for every $i = 0, \dots, r_\ell$, $Y_i \xrightarrow{\ell} \{Y'_i, Y''_i\}$;
 1109 b. for every $i = 1, \dots, r_\ell$, $Y_i = Z_{i-1} = X_{i+1}$;

1110 By construction, $(G, H^*, \vec{B}^*, \mathbf{h}^*(G))$ produced by encoding the unfolding of the embedding
 1111 of G on \mathbb{T}_k , described in Section 3.2.2, is locally consistent. The following result shows that
 1112 the local notion of historical consistency based on the histories fits with the global notion of
 1113 historical consistency used in Section 3.2.2.

1114 ► **Lemma 14.** *Let H be a splitting of a graph G , let f be a planar embedding of H , let ϕ be*
 1115 *a face of H with boundary walk \vec{B} directed clockwise. Let $\mathbf{h}(G)$ be a history of all the vertices*
 1116 *in G . If $(G, H, \vec{B}, \mathbf{h}(G))$ is locally consistent, then $(G, H, \vec{B}, \mathcal{U})$ is globally consistent, where*
 1117 *$\mathcal{U} = U_0, \dots, U_{3k-1}$ is a sequence of graphs enabling Condition 1 of the historical consistency*
 1118 *of $(G, H, \vec{B}, \mathbf{h}(G))$ to hold.*

1119 **Proof.** Thanks to Condition 1, for every $0 \leq \ell < 3k - 1$, $U_{\ell+1}$ is a degree-2 splitting of U_ℓ .
 1120 Moreover, by the consistence of the footprints and the typing in the histories, the splitting
 1121 of from U_ℓ to $U_{\ell+1}$ is locally consistent at each node of U_i with the duplication of a cycle
 1122 whenever $\ell \leq k$, and with the duplication of a path otherwise.

1123 Condition 2 in the definition of local consistency guarantees that the footprints at the
 1124 leaves of the histories are correctly set, that is, they collectively encode the boundary walk B .

1125 Condition 3 guarantees that, for $\ell = 1, \dots, 2k - 1$, starting from $\chi_{2k} = B$, one can
 1126 iteratively decompose the boundary walk of the face $\chi_{\ell+1}$ of $U_{k+\ell+1}$ into a boundary walk
 1127 of a face $\psi_{\ell+1}$ of $U_{k+\ell}$, a boundary walk of a face χ_ℓ of $U_{k+\ell}$, and the duplication of a
 1128 path in $U_{k+\ell}$ connecting χ_ℓ to $\psi_{\ell+1}$. It follows that $2k$ faces ψ_1, \dots, ψ_{2k} of U_ℓ have been
 1129 identified. Since, the merging of the $2k - 1$ paths successively identified in the graphs $U_{k+\ell}$,
 1130 $\ell = 1, \dots, 2k - 1$ preserves planarity, the graph U_k is planar.

1131 Moreover, each of the boundary walks of the faces ψ_1, \dots, ψ_{2k} is oriented in a direction
 1132 inherited from the clockwise orientation of B , as guaranteed by the Elementary, Extremity,
 1133 and Vacancy rules satisfied by the footprints, whose validity are themselves guaranteed by
 1134 Condition 1. Condition 4 guarantees that the $2k$ faces ψ_1, \dots, ψ_{2k} of U_k can be reordered as
 1135 k pairs (ϕ'_i, ϕ''_i) , $i \in \{1, \dots, k\}$ that can be successively merged for creating handles. More
 1136 specifically, for $i = k, k - 1, \dots, 1$, Condition 4 guarantees that the boundary walks of ϕ'_i and
 1137 ϕ''_i are directed such that, by identifying the vertices of U_i that are split of vertices in U_{i-1} ,
 1138 a handle is created, resulting in U_i embedded in \mathbb{T}_{k-i} . ◀

1139 4.5 Existence and Unicity of the Paths and Cycles

1140 Our proof-labeling scheme relies on a collection of paths and cycles in the graphs
 1141 $G^{(0)}, \dots, G^{(3k-1)}$. The footprints and types encode these paths and cycles locally. One
 1142 needs to guarantee the existence and unicity of each path and cycle, in each graph $G^{(i)}$,
 1143 $i = 0, \dots, 3k - 1$. The next lemma, which is standard, achieve this task.

1144 ▶ **Lemma 15.** *Let G be a graph, and let P (resp., C) be a (non-necessary simple) directed path
 1145 (resp., cycle) in G . Assume each vertex v of P (resp., C) is given a triple $(\text{pred}(v), v, \text{succ}(v))$,
 1146 where $\text{pred}(v)$ and $\text{succ}(v)$ are the predecessor and successor of v in P (resp., C). If v is an
 1147 extremity of P , then $\text{pred}(v) = \perp$ or $\text{succ}(v) = \perp$, or both $\text{pred}(v) = \perp$ and $\text{succ}(v) = \perp$ in
 1148 case P is reduced to v . There exists a proof-labeling scheme with certificates on $O(\log n)$ bits
 1149 that guarantees the existence and unicity of P .*

1150 **Proof.** Let P be a directed path in G . The proof-labeling scheme uses a spanning tree T of
 1151 G rooted at the starting vertex v_0 of P . Every vertex v is given the ID of its parent $p(v)$
 1152 in T (v_0 has $p(v_0) = \perp$). The tree T is certified by providing a certificate to every node v
 1153 containing a pair $(\text{id}(v_0), d(v))$, where $d(v)$ is the distance from v to v_0 in T . Every vertex v
 1154 checks that it is given the same root-ID as its neighbors in G , and that $d(p(v)) = d(v) - 1$.
 1155 Every node that is given one or many triples $(\text{pred}(v), v, \text{succ}(v))$ checks that, for each of
 1156 them, $\text{pred}(\text{succ}(v)) = v$ and $\text{succ}(\text{pred}(v)) = v$. (Of course, every such vertex v also checks
 1157 consistence of the triples given to it, including the fact that $\text{pred}(v) \neq \text{succ}(v)$ unless they
 1158 are both equal to \perp , that it is not given the same successor in two different triples, etc.). If
 1159 one of the tests is not passed at a vertex, this vertex rejects, otherwise it accepts. The case
 1160 of a cycle C is treated the same, where the spanning tree T is rooted at any vertex of C . It
 1161 is easy to check that this standard proof-labeling scheme satisfies both completeness and
 1162 soundness. ◀

1163 4.6 Verification Procedure

1164 We now have all ingredients for describing our proof-labeling scheme for \mathcal{G}_k^+ , $k \geq 0$. First, we
 1165 describe the certificates assigned to the vertices of a graph G of genus k . The main part of the
 1166 certificate of v is the history $h(v)$, as constructed in Section 4.1. As mentioned in Section 4.2,

1167 a history may require more than just $O(\log n)$ bits. However, Lemma 12 has shown how to
 1168 resolve this issue, so that histories can be spread out among the vertices in a way guaranteeing
 1169 that every vertex stores $O(\log n)$ bits, and, in a single round of communication with its
 1170 neighbors, every node v can recover its entire history. More importantly even, although a
 1171 vertex v may not be able to recover the whole history of each of its neighbors in a single
 1172 round, yet it can recover from each neighbor w the part of $h(w)$ corresponding to every edge
 1173 between an avatar of v and an avatar of w , which is sufficient to check the consistency of the
 1174 neighborhoods, footprints, etc., in all graphs $G^{(0)}, \dots, G^{(3k-1)}$ used in the construction. In
 1175 addition, the certificate of every vertex is provided with the information enabling to check
 1176 planarity of $H = G^{(3k-1)}$ (cf. Lemma 13), and to guarantee the existence and unicity of all
 1177 the directed cycles $C'_i, C''_i, i = 1, \dots, k$, and all directed paths $P'_j, P''_j, j = 1, \dots, 2k - 1$ (cf.
 1178 Lemma 15). The vertices can then check local consistency, as specified in Section 4.4. Since
 1179 G has genus k , it follows that, whenever the prover assigns the certificates appropriately, all
 1180 vertices pass all tests, and therefore all vertices accept. Completeness is therefore satisfied by
 1181 the scheme.

1182 Soundness is guaranteed by Lemmas 10 and 14. Indeed, the latter lemma shows that if
 1183 the vertices are given certificates that are consistent, and in particular for which the histories
 1184 are locally consistent, then global consistency is also guaranteed. And the former lemma
 1185 says that if global consistency is satisfied then the graph can be embedded on \mathbb{T}_k . Therefore,
 1186 if a graph G cannot be embedded on \mathbb{T}_k , then global consistency cannot be satisfied, which
 1187 means that the local consistency of the histories cannot be satisfied either, and therefore,
 1188 at least one vertex of G fails to pass all tests, and rejects. This completes the proof of
 1189 Theorem 11.

1190 **5 Proof-Labeling Scheme for Bounded Non-orientable genus Graphs**

1191 This section is entirely dedicated to the proof of our second main result.

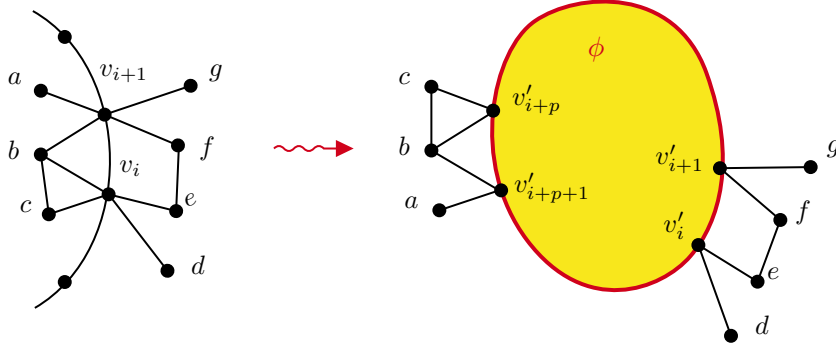
1192 ► **Theorem 16.** *Let $k \geq 0$, and let \mathcal{G}_k^- be the class of graphs with non-orientable genus at*
 1193 *most k , i.e., embeddable on a non-orientable closed surface of genus at most k . There is a*
 1194 *proof-labeling scheme for \mathcal{G}_k^- using certificates on $O(\log n)$ bits in n -node graphs.*

1195 The proof-labeling scheme for \mathcal{G}_k^- is based on the same ingredients as the one for
 1196 \mathcal{G}^+ in Theorem 11 (e.g., Lemma 4 is used in replacement of Lemma 3, etc.). However,
 1197 new ingredients must be introduced for handling the cross-caps from which non-orientable
 1198 surfaces result. The proof will thus mainly consist in describing these new ingredients, and
 1199 in explaining their interactions with the ingredients used for establishing Theorem 11. We
 1200 start by defining the notion of *doubling* performed on cycles.

1201 **5.1 Doubling of a Non-Orientable Cycle**

1202 Let us assume that we are given an embedding of a graph G on a non-orientable closed
 1203 surface Σ of genus k , and let $D = (v_0, v_1, \dots, v_{p-1}, v_p = v_0)$ be a non-orientable cycle of G .
 1204 Note that a non-orientable cycle is non-separating. The graph G_D is obtained by *doubling* D ,
 1205 i.e., by multiplying its length by 2. This doubling of D , and the canonical embedding of G_D
 1206 on a closed surface Σ_D , are obtained as follows (see Figure 17 for an illustration).

1207 ■ Each vertex $v_i, 0 \leq i < p$, is split into two vertices v'_i and v'_{p+i} in such a way that
 1208 $D' = (v'_0, v'_1, v'_2, \dots, v'_{2p-1}, v'_{2p} = v'_0)$ is a cycle of G_D , which forms a boundary walk of a
 1209 face ϕ of X_D .



■ **Figure 17** Doubling a non-orientable cycle.

- 1210 ■ The neighbors of each vertex v_i in $G \setminus D$, $0 \leq i < p$, are shared between v'_i and v'_{i+p} in G_D ,
 1211 as follows. The left and right sides of D can be defined locally, i.e., in the neighborhood
 1212 of each (embedded) edge $\{v_i, v_{i+1}\}$ of D . The edges incident to v'_i and v'_{i+1} in G_D (and,
 1213 by symmetry, the edges incident to v'_{i+p} and v'_{i+p+1}) correspond to the edges incident to
 1214 v_i and v_{i+1} on the same side of D in G according to the local definition of left and right
 1215 sides in the neighborhood of $\{v_i, v_{i+1}\}$.
 1216 ■ The vertices v'_i and v'_{i+p} have no other neighbors.

1217 We now show how to unfold \mathbb{P}_k , as we did for unfolding \mathbb{T}_k in the oriented case.

1218 5.2 Unfolding \mathbb{P}_k for $k \geq 1$

1219 Let G be a graph with a 2-cell embedding f on \mathbb{P}_k . The unfolding of G has three phases,
 1220 and only the first one, called *doubling phase* is new. The second phase is a face-duplication
 1221 phase, and the third phase is a face-reduction phase, identical to those described in the case
 1222 of orientable surfaces. The doubling phase is as follows. Let $\Sigma^{(0)} = \mathbb{P}_k$, and let D_1 be a
 1223 non-orientable cycle of $G^{(0)} = G$. Let us consider the embedding of $G^{(1)} = G_{D_1}^{(0)}$ induced
 1224 by f , on the surface $\Sigma^{(1)} = \Sigma_{D_1}^{(0)}$. There are two cases, both using Lemma 5:

- 1225 ■ If $\Sigma^{(1)}$ is non-orientable, then $\Sigma^{(1)}$ is homeomorphic to \mathbb{P}_{k-1} ;
 1226 ■ Otherwise, $\Sigma^{(1)}$ is homeomorphic to $\mathbb{T}_{\frac{k-1}{2}}$.

1227 In the first case, a doubling operation is repeated on $G^{(1)}$, using a non-orientable cycle D_2 of
 1228 $G^{(1)}$. Doubling operations are performed iteratively until an embedding on an orientable
 1229 surface is reached. Formally, there exists a sequence of $m+1$ graphs $G^{(0)}, \dots, G^{(m)}$, $m \leq k$,
 1230 respectively embedded on closed surfaces $\Sigma^{(0)}, \dots, \Sigma^{(m)}$, such that, for $0 \leq i < m$, there exists
 1231 a non-orientable cycle D_{i+1} of $G^{(i)}$ such that $G^{(i+1)} = (G^{(i)})_{D_{i+1}}$, and $\Sigma^{(i+1)} = (\Sigma^{(i)})_{D_{i+1}}$
 1232 (up to homeomorphism). Necessarily, for $0 \leq i < m$, $\Sigma^{(i)} = \mathbb{P}_{k-i}$ (up to homeomorphism),
 1233 and $\Sigma^{(m)} = \mathbb{T}_{(k-m)/2}$, thanks to Lemma 5. When $\Sigma^{(m)}$ is reached, $G^{(m)}$ contains m special
 1234 faces, whose boundary walks are resulting from the successive doubling of D_1, \dots, D_m ,
 1235 respectively. The doubling phase is then completed.

1236 The face duplication phase starts, initialized with the embedding of $G^{(m)}$ on $\Sigma^{(m)}$. Let
 1237 $k' = \frac{k-m}{2}$. The duplication phase is performed, as in Section 3.2.3. Specifically, there exists
 1238 a sequence of $k'+1$ graphs $G^{(m)}, \dots, G^{(m)+k'}$, respectively embedded on closed surfaces
 1239 $\Sigma^{(m)}, \dots, \Sigma^{(m)+k'}$, such that, for $0 \leq i < k'$, there exists a non-separating cycle C_{i+1} of
 1240 $G^{(m+i)}$ such that $G^{(m+i+1)} = G_{C_{i+1}}^{(m+i)}$, and $\Sigma^{(m+i+1)} = \Sigma_{C_{i+1}}^{(m+i)}$. Necessarily, for $0 \leq i \leq k'$,
 1241 $\Sigma^{(m+i)} = \mathbb{T}_{k'-i}$ up to homeomorphism, thanks to Lemma 5. In particular, $\Sigma^{(m+k')} = \mathbb{T}_0$.

1242 When $\Sigma^{(m+k')}$ is reached, $G^{(m+k')}$ contains $2k' + m$ special faces, whose boundary walks are
 1243 resulting from the successive doubling of the cycles D_1, \dots, D_m , and from the duplications
 1244 of the cycles $C_1, \dots, C_{k'}$. At this point, the face-duplication phase is completed.

1245 The face-reduction phase starts, as in Section 3.2.4, in order to merge the $2k' + m = k$
 1246 special faces of $G^{(m+k')}$ into a single face. Let us denote the $2k' + m = k$ special faces of
 1247 $G^{(m+k')}$ by ψ_1, \dots, ψ_k . Let $\psi_1 = \chi_1$. There exists a sequence of paths P_1, \dots, P_{k-1} such
 1248 that, for $1 \leq i \leq k-1$, the duplication of P_i merges χ_i and ψ_{i+1} in a single face χ_{i+1} . A
 1249 sequence of planar graphs $G^{(m+k')}, \dots, G^{(m+k'+k-1)}$ results from these merges, where, for
 1250 $0 \leq i < k-1$, P_{i+1} is a path of $G^{(m+k'+i)}$, and $G^{(m+k'+i+1)} = G_{P_{i+1}}^{(m+k'+i)}$. For $1 \leq i \leq k-1$,
 1251 $G^{(m+k'+i)}$ has $k-i$ special faces $\chi_{i+1}, \psi_{i+2}, \dots, \psi_k$. In particular, $G^{(m+k'+k-1)}$ has a unique
 1252 special face χ_{k-1} .

1253 To summarize, as in Section 3.2.2, the embedding f of G in \mathbb{P}_k induces a planar embedding
 1254 of $H^* = G^{(m+k'+k-1)}$ whose external face is $\phi^* = \chi_{k-1}$. The boundary of face ϕ^* contains
 1255 all the vertices obtained by splittings resulting from doublings or duplications.

1256 5.3 Certifying Non-Orientable Genus at Most k

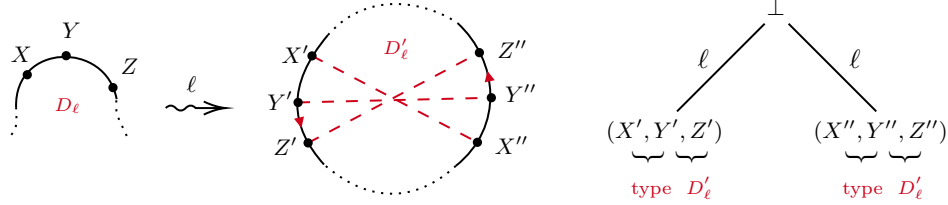
1257 Conversely, for a graph G of non-orientable genus k , an embedding of G in \mathbb{P}_k can be induced
 1258 from the embedding f^* of H^* on \mathbb{T}_0 , and from the boundary walk B^* of ϕ^* . The latter is
 1259 indeed entirely determined by the successive cycle-duplications, path-duplications, and cycle
 1260 doublings performed during the whole process. It contains all duplicated vertices resulting
 1261 from the cycles D'_1, \dots, D'_m , the cycles $C'_1, \dots, C'_{k'}$ and $C''_1, \dots, C''_{k'}$, and from the paths
 1262 P'_1, \dots, P'_{k-1} and P''_1, \dots, P''_{k-1} .

1263 Now, let H be a splitting of a graph G , let f be a planar embedding of H , and let ϕ
 1264 be a face of H embedded on \mathbb{T}_0 . Let $B = (u_0, \dots, u_N)$ be a boundary walk of ϕ , and let
 1265 \vec{B} be an arbitrary direction given to B , say clockwise. Let $\mathcal{U} = (U_0, \dots, U_{m+k'+k-1})$, with
 1266 $m + 2k' = k$ and $m \geq 1$, be a sequence of graphs such that $U_0 = G$, $U_{m+k'+k-1} = H$, and,
 1267 for every $i \in \{0, \dots, m+k'+k-1\}$, U_{i+1} is a 2-splitting of U_i . The splitting of U_i into
 1268 U_{i+1} is denoted by $\sigma_i = (\alpha_i, \beta_i)$. The definition of global consistency of $(G, H, \vec{B}, \mathcal{U})$, in the
 1269 case of orientable surfaces, can trivially be adapted to the case of non-orientable surfaces
 1270 by revisiting conditions 1 and 2, of Section 3.2.5, in such a way that the indices correspond
 1271 to the unfolding of \mathbb{P}_k . We thus say that $(G, H, \vec{B}, \mathcal{U})$ is *globally consistent* for \mathbb{P}_k if the
 1272 (revisited) conditions 1 and 2 in Section 3.2.5 hold, plus the following additional condition
 1273 corresponding to the doubling phase:

- 1274 ■ **Cycle doubling checking.** For every $i = 1, \dots, \ell$, there exist faces $\phi_1^{(i)}, \phi_2^{(i)}, \dots, \phi_i^{(i)}$
 1275 of U_i with respective directed boundary walks $\vec{B}(\phi_1^{(i)}), \vec{B}(\phi_2^{(i)}), \dots, \vec{B}(\phi_i^{(i)})$ such that
 1276 ■ $\vec{B}(\phi_i^{(i)}) = (v'_0, v'_1, \dots, v'_{2p-1}, v'_{2p} = v'_0)$ with, for $0 \leq j < p$, $\sigma_{i-1}^{-1}(\{v'_j, v'_{j+p}\}) \in$
 1277 $V(U_{i-1})$;
 1278 ■ for $j = 1, \dots, i-1$, $\sigma_{i-1}(\vec{B}(\phi_j^{(i-1)})) = \vec{B}(\phi_j^{(i)})$.

1279 By the construction of Section 5.2, for every graph G of non-orientable genus k ,
 1280 $(G, H^*, \vec{B}^*, \mathcal{U}^*)$ is globally consistent for \mathbb{P}_k , where $\mathcal{U}^* = (G^{(0)}, \dots, G^{(m+k'+k-1)})$. The
 1281 following lemma is the analog to Lemma 10 for non-orientable surface. Its proof is essentially
 1282 the same as the proof of Lemma 10, in which an argument should be added, for handling
 1283 cycle doublings, that is, for identifying opposite vertices of the cycle D'_i in order to create a
 1284 cross-cap. The details are omitted.

1285 ► **Lemma 17.** *Let H be a splitting of a graph G , and assume that there exists a planar*
 1286 *embedding f of H with a face ϕ and a boundary walk B of ϕ . Let m, k' be integers such that*



■ **Figure 18** The cross-cap rule.

1287 $1 \leq m \leq k$ and $m + 2k' = k$, and let $\mathcal{U} = (U_0, \dots, U_{m+k'+k-1})$ be a series of graphs such that
 1288 $U_0 = G$, $U_{m+k'+k-1} = H$, and, for every $i \in \{0, \dots, m+k'+k-2\}$, U_{i+1} is a 2-splitting of
 1289 U_i . If $(G, H, \vec{B}, \mathcal{U})$ is globally consistent for \mathbb{P}_k , then G can be embedded on \mathbb{P}_k .

1290 Thanks to Lemma 17, the overall outcome of this section is that the tuple $c =$
 1291 $(H^*, f^*, \phi^*, B^*, \mathcal{U}^*)$ constructed in Section 5.2 is indeed a certificate that G can be em-
 1292 bedded on \mathbb{P}_k .

1293 5.4 From Centralized Certificate to Local Certificate

The method to distribute the centralized certificates uses the same approach and the same tools as those used in Section 4 in the orientable case. Only the differences are pointed out in this section. In the non-orientable case, the set of types is

$$S_k = \{D'_1, \dots, D'_\ell, C'_1, \dots, C'_{k'}, C''_1, \dots, C''_{k'}, P'_1, \dots, P'_{k-1}, P''_1, \dots, P''_{k-1}\}.$$

1294 The footprints and their construction are identical to the orientable case, except that a
 1295 cross-cap rule is introduced (see Figure 18).

1296 **Cross-cap rule.** Assuming $X \xrightarrow{\ell} X', X''$, $Y \xrightarrow{\ell} Y', Y''$, and $Z \xrightarrow{\ell} Z', Z''$, the cross-cap rule
 1297 matches two footprints of two children Y' and Y'' , and produces none at the parent Y :

$$1298 (X', Y', Z'), (X'', Y'', Z'') \xrightarrow{\ell} \perp.$$

1299 The cross-cap rule applies to the case of identifying opposite vertices of the boundary of
 1300 a face, in the reverse operation of doubling. The corresponding face disappears, and their
 1301 boundaries can be discarded.

1302 The assignments of types to footprints is performed in the same as in Section 4, and the
 1303 same distributed algorithm is used for checking the planarity of H . An important difference
 1304 with the orientable case appears in the definition of the local consistency of distributed
 1305 certificates (previously defined in Section 4.4). Again, an additional condition is introduced,
 1306 for reflecting the creation of cross-caps.

1307 ■ For every $\ell = 1, \dots, m$, the collection of footprints at the nodes at level ℓ whose
 1308 both edges have type D'_ℓ in the histories in $\mathbf{h}(G)$ can be ordered as $(X'_0, Y'_0, Z'_0), \dots,$
 1309 $(X'_{2r_\ell-1}, Y'_{2r_\ell-1}, Z'_{2r_\ell-1})$, for some $r_\ell \geq 1$, such that:

- 1310 1. for every $i = 0, \dots, r_\ell - 1$, $Y_i \xrightarrow{\ell} \{Y'_i, Y''_{i+r_\ell}\}$;
- 1311 2. for every $i = 0, \dots, 2r_\ell - 1$, $Y_i = Z_{i-1} = X_{i+1}$ (where indices are taken modulo $2r_\ell$);

1312 The following lemma is the analog of Lemma 14, but for non-orientable surfaces. Its proof is
 1313 identical to the proof of Lemma 14, with an additional argument, stating that the conditions
 1314 added for handling non-orientable surfaces enable opposite vertices of the face surrounded by
 1315 D'_ℓ in U_ℓ , $1 \leq \ell \leq 2k - 1$, to be identified for creating a cross-cap in $U_{\ell-1}$.

1316 ► **Lemma 18.** *Let H be a splitting of a graph G , let f be a planar embedding of H , let ϕ*
 1317 *be a face of H with boundary walk \vec{B} directed clockwise. Let $h(G)$ be a history of all the*
 1318 *vertices in G . If $(G, H, \vec{B}, h(G))$ is locally consistent, then $(G, H, \vec{B}, \mathcal{U})$ is globally consistent,*
 1319 *where $\mathcal{U} = U_0, \dots, U_{m+k'+k-1}$ is a sequence of graphs enabling the global consistency of*
 1320 *$(G, H, \vec{B}, h(G))$ to hold.*

1321 5.5 Verification Procedure

1322 The verification procedure is similar to the one described in Section 4.6, and is therefore
 1323 omitted.

1324 6 Conclusion

1325 In this paper, we have designed proof-labeling schemes for the class of graphs of bounded
 1326 genus, as well as for the class of graphs with bounded non-orientable genus. All our schemes
 1327 use certificates on $O(\log n)$ bits, which is optimal, as it is known that even certifying
 1328 the class of planar graphs requires proof-labeling schemes with certificates on $\Omega(\log n)$
 1329 bits [21]. The existence of “compact” proof-labeling schemes (i.e., schemes using certificates
 1330 of polylogarithmic size) for other classes of sparse graphs is still not known. In particular,
 1331 proving or disproving the existence of such a scheme for H -minor-free graphs appears to
 1332 be a challenging problem. Indeed, Robertson and Seymour’s decomposition theorem states
 1333 that every H -minor-free graph can be expressed as a tree structure of “pieces”, where each
 1334 piece is a graph that can be embedded in a surface on which H cannot be embedded, plus
 1335 a bounded number of so-called *apex* vertices, and a bounded number of so-called *vortex*
 1336 subgraphs. The decomposition theorem provides a powerful tool for the design of (centralized
 1337 or distributed) algorithms. However, this theorem is not a characterization, that is, there are
 1338 graphs that are not H -minor-free, and yet can be expressed as a tree structure satisfying the
 1339 required properties (surfaces of bounded genus, bounded number of apices, bounded number
 1340 of vortices, etc.). It follows that, although Robertson and Seymour’s decomposition theorem
 1341 should most probably play a crucial role for designing a compact proof-labeling scheme for
 1342 H -minor-free graphs (if such a scheme exists), this development may require identifying
 1343 additional properties satisfied by these graphs.

1344 — References —

- 1345 1 Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Compact routing for graphs excluding a
 1346 fixed minor. In *19th International Conference on Distributed Computing (DISC)*, LNCS 3724,
 1347 pages 442–456. Springer, 2005.
- 1348 2 Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its application
 1349 to self-stabilization. *Theor. Comput. Sci.*, 186(1-2):199–229, 1997.
- 1350 3 Saeed Akhoondian Amiri, Patrice Ossona de Mendez, Roman Rabinovich, and Sebastian
 1351 Siebertz. Distributed domination on graph classes of bounded expansion. In *30th ACM
 1352 Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 143–151, 2018.
- 1353 4 Saeed Akhoondian Amiri, Stefan Schmid, and Sebastian Siebertz. A local constant factor MDS
 1354 approximation for bounded genus graphs. In *ACM Symposium on Principles of Distributed
 1355 Computing (PODC)*, pages 227–233, 2016.
- 1356 5 Saeed Akhoondian Amiri, Stefan Schmid, and Sebastian Siebertz. Distributed dominating set
 1357 approximations beyond planar graphs. *ACM Trans. Algorithms*, 15(3):39:1–39:18, 2019.
- 1358 6 Baruch Awerbuch, Boaz Patt-Shamir, and George Varghese. Self-stabilization by local checking
 1359 and correction (extended abstract). In *32nd Symposium on Foundations of Computer Science
 1360 (FOCS)*, pages 268–277, 1991.

- 1361 7 Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. What can be
1362 verified locally? *J. Comput. Syst. Sci.*, 97:106–120, 2018.
- 1363 8 Marthe Bonamy, Cyril Gavoille, and Michal Pilipczuk. Shorter labeling schemes for planar
1364 graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 446–462. SIAM,
1365 2020.
- 1366 9 H. R. Brahana. Systems of circuits on two-dimensional manifolds. *Annals of Mathematics*,
1367 23:144–168, 1922.
- 1368 10 Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theor.*
1369 *Comput. Sci.*, 811:112–124, 2020.
- 1370 11 Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive
1371 proofs. In *33rd International Symposium on Distributed Computing (DISC)*, LIPIcs 146, pages
1372 13:1–13:17. Dagstuhl, 2019.
- 1373 12 Andrzej Czygrinow and Michał Hańćkowiak. Distributed almost exact approximations for
1374 minor-closed families. In *14th Annual European Symposium on Algorithms (ESA)*, pages
1375 244–255, 2006.
- 1376 13 Andrzej Czygrinow, Michał Hańćkowiak, Edyta Szymanska, Wojciech Wawrzyniak, and Marcin
1377 Witkowski. Distributed local approximation of the minimum k-tuple dominating set in planar
1378 graphs. In *18th Int. Conference on Principles of Distributed Systems (OPODIS)*, pages 49–59,
1379 2014.
- 1380 14 Andrzej Czygrinow, Michał Hańćkowiak, Edyta Szymanska, Wojciech Wawrzyniak, and Marcin
1381 Witkowski. Improved distributed local approximation algorithm for minimum 2-dominating
1382 set in planar graphs. *Theor. Comput. Sci.*, 662:1–8, 2017.
- 1383 15 Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approxi-
1384 mations in planar graphs. In *22nd Int. Symp. on Distributed Computing (DISC)*, pages 78–92,
1385 2008.
- 1386 16 Vida Dujmovic, Louis Esperet, Cyril Gavoille, Gwenaél Joret, Piotr Micek, and Pat Morin.
1387 Adjacency labelling for planar graphs (and beyond). In *61st IEEE Symposium on Foundations
1388 of Computer Science (FOCS)*, pages 577–588, 2020.
- 1389 17 Louis Esperet and Benjamin Lévêque. Local certification of graphs on surfaces. *CoRR*,
1390 abs/2102.04133, Feb 8, 2021.
- 1391 18 Laurent Feuilloley. Bibliography of distributed approximation beyond bounded degree. *CoRR*,
1392 abs/2001.08510, 2020.
- 1393 19 Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A hierarchy of local decision. *Theor.*
1394 *Comput. Sci.*, 856:51–67, 2021.
- 1395 20 Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, and Mor Perry. Redundancy
1396 in distributed proofs. *Distributed Computing*, page To appear, 2021.
- 1397 21 Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and
1398 Ioan Todinca. Compact distributed certification of planar graphs. In *39th ACM Symposium
1399 on Principles of Distributed Computing (PODC)*, pages 319–328, 2020.
- 1400 22 Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local
1401 distributed computing. *J. ACM*, 60(5):35:1–35:26, 2013.
- 1402 23 Pierre Fraigniaud, Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. On
1403 distributed Merlin-Arthur decision protocols. In *26th Int. Colloquium Structural Information
1404 and Communication Complexity (SIROCCO)*, LNCS 11639, pages 230–245. Springer, 2019.
- 1405 24 Pierre Fraigniaud, Boaz Patt-Shamir, and Mor Perry. Randomized proof-labeling schemes.
1406 *Distributed Computing*, 32(3):217–234, 2019.
- 1407 25 Cyril Gavoille and Nicolas Hanusse. Compact routing tables for graphs of bounded genus.
1408 In *26th Int. Coll. on Automata, Languages and Programming (ICALP)*, LNCS 1644, pages
1409 351–360. Springer, 1999.
- 1410 26 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks I: planar
1411 embedding. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages
1412 29–38, 2016.

- 1413 27 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II:
1414 low-congestion shortcuts, MST, and min-cut. In *27th ACM-SIAM Symposium on Discrete*
1415 *Algorithms (SODA)*, pages 202–219, 2016.
- 1416 28 Mohsen Ghaffari and Merav Parter. Near-optimal distributed DFS in planar graphs. In *31st*
1417 *Int. Symp. on Distributed Computing (DISC)*, LIPIcs, pages 21:1–21:16. Dagstuhl, 2017.
- 1418 29 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of*
1419 *Computing*, 12(1):1–33, 2016.
- 1420 30 Miikka Hilke, Christoph Lenzen, and Jukka Suomela. Brief announcement: local approxima-
1421 bility of minimum dominating set on planar graphs. In *ACM Symposium on Principles of*
1422 *Distributed Computing (PODC)*, pages 344–346, 2014.
- 1423 31 Piotr Indyk and Anastasios Sidiropoulos. Probabilistic embeddings of bounded genus graphs
1424 into planar graphs. In Jeff Erickson, editor, *Proceedings of the 23rd ACM Symposium on*
1425 *Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 204–209. ACM, 2007.
- 1426 32 Gene Itkis and Leonid A. Levin. Fast and lean self-stabilizing asynchronous protocols. In *35th*
1427 *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 226–239, 1994.
- 1428 33 Gillat Kol, Rotem Oshman, and Raghuvansh R. Saxena. Interactive distributed proofs. In
1429 *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 255–264, 2018.
- 1430 34 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*,
1431 22(4):215–233, 2010.
- 1432 35 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed
1433 locally! In *23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages
1434 300–309, 2004.
- 1435 36 Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated
1436 locally?: case study: dominating sets in planar graphs. In *20th ACM Symposium on Parallelism*
1437 *in Algorithms and Architectures (SPAA)*, pages 46–54, 2008.
- 1438 37 Christoph Lenzen, Yvonne Anne Pignolet, and Roger Wattenhofer. Distributed minimum
1439 dominating set approximations in restricted families of graphs. *Distributed Computing*,
1440 26(2):119–137, 2013.
- 1441 38 W.S. Massey, J.H. Ewing, F.W. Gerhing, and P.R. Halmos. *A Basic Course in Algebraic*
1442 *Topology*. Graduate Texts in Mathematics. Springer New York, 1991.
- 1443 39 B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins Studies in the Mathematical
1444 Sciences. Johns Hopkins University Press, 2001.
- 1445 40 Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive
1446 proofs. In *31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–1115,
1447 2020.
- 1448 41 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*,
1449 24(6):1259–1277, 1995.
- 1450 42 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*,
1451 volume 28 of *Algorithms and combinatorics*. Springer, 2012.
- 1452 43 Ronald Ortner. Embeddability of arrangements of pseudocircles into the sphere. *European*
1453 *Journal of Combinatorics*, 29(2):457–469, 2008.
- 1454 44 Torrence D. Parsons, Giustina Pica, Tomaz Pisanski, and Aldo G. S. Ventre. Orientably
1455 simple graphs. *Mathematica Slovaca*, 37(4):391–394, 1987.
- 1456 45 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.
- 1457 46 David Peleg and Vitaly Rubinfeld. A near-tight lower bound on the time complexity of
1458 distributed minimum-weight spanning tree construction. *SIAM J. Comput.*, 30(5):1427–1442,
1459 2000.
- 1460 47 Henri Poincaré. Sur la généralisation d’un théorème d’Euler relatif aux polyèdres. *C.R. Hebdo.*
1461 *Séances Académie des Sciences*, 117:144–145, 1893.
- 1462 48 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal
1463 Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of
1464 distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

- 1465 49 Wojciech Wawrzyniak. A strengthened analysis of a local algorithm for the minimum domi-
1466 nating set problem in planar graphs. *Inf. Process. Lett.*, 114(3):94–98, 2014.
- 1467 50 Wojciech Wawrzyniak. A local approximation algorithm for minimum dominating set problem
1468 in anonymous planar networks. *Distributed Computing*, 28(5):321–331, 2015.
- 1469 51 J. W. T. Youngs. Minimal imbeddings and the genus of a graph. *Journal of Mathematical*
1470 *Mechanics*, 12:303–315, 1963.