COM

COMPLEXITY COLUMN

NEIL IMMERMAN, University of Massachusetts Amherst
immerman@cs.umass.edu

Some of my favorite open problems concern fixed-point logic with counting, FPC. It is known that counting logic with $k+1$ variables, C^{k+1}, has exactly the same expressive power as the classic k-dimensional Weisfeiler-Leman Algorithm, k-WL. Furthermore, the quantifier-depth of a C^{k+1} formula needed to express the color of a k-tuple of vertices is equal to the number of iterations of k-WL needed to derive that color.

Much has been learned about FPC and the power of k-WL in the last forty years; but the most important mysteries remain. Sandra Kiefer's column guides us through some of the beautiful new results in her thesis. In particular, she settles to within one the number of variables needed to characterize planar graphs in counting logic: either 3 or 4. The best previously known bound was that 15 variables suffice. She also sketches the proof of a generalization of this result, showing that the number of variables needed to characterize graphs of genus, g, is at most $4g+4$.

The Weisfeiler-Leman Algorithm: An Exploration of its Power

Sandra Kiefer, RWTH Aachen University

More than half a century after its first formulation, the Weisfeiler-Leman (WL) algorithm is still an important combinatorial technique whenever graphs or other relational structures are to be classified. However, despite its simple algebraic description and its variety of applications, we still lack a precise understanding of the expressive power of the algorithm.

This column introduces the reader to the basic concepts of the WL algorithm and discusses its dimension as a parameter to capture the structural complexity of an input graph. Specifically, I present a survey of work regarding the WL dimension conducted with my co-authors. First, I outline the proof that the 3-dimensional WL algorithm (3-WL) is able to identify every planar graph. The proof version presented here relies on strong insights about the ability of 2-WL to decompose graphs. Afterwards, I highlight the most important ingredients of the generalisation of our bound to graphs that are parameterised by their Euler genus.

Further details as well as a study of other aspects of the WL algorithm can be found in my dissertation [Kiefer 2020].

1. INTRODUCTION

The Colour Refinement procedure and its generalisation to higher dimensions, the Weisfeiler-Leman (WL) algorithm, are powerful combinatorial tools to classify graphs and other relational structures. The algorithm first appeared in the literature more than half a century ago (see, for instance, [Morgan 1965]), but, as very recent publications show, it is still widely used and studied both in theoretical and practical computer science (see, for example, [Fuhlbrück et al. 2020; Grohe et al. 2020]). Most notably, Babai employs a high-dimensional variant of it in his quasipolynomial-time graph isomorphism algorithm [Babai 2016].

The general concept behind the WL algorithm is simple and intuitive: given an input graph, assign colours to the vertices that encode, in an isomorphism-invariant way, structural information about the roles of the vertices in the surrounding graph. These colours are computed in iterations, taking into account previously computed colours. Considering this very natural approach to classifying vertices in graphs, it may come as a surprise that there are still numerous open questions regarding the power of the algorithm.

Persistent interest in it has led to the discovery of many links of the WL algorithm to seemingly unrelated areas. For example, there is a precise correspondence between the algorithm and a fragment of first-order logic with counting as well as a certain type of Ehrenfeucht-Fraïssé game. Also, a recently discovered connection to machine learning shows that the Colour Refinement procedure captures graph neural networks with respect to their expressive power [Morris et al. 2019].

In the graph isomorphism context, the algorithm is usually applied to two input graphs in parallel in order to decide whether they are isomorphic. Distinct colours of two vertices imply that there is no isomorphism mapping one to the other. Thus, a colour occuring in the first graph but not in the second implies that they are not isomorphic. Therefore, the graphs are always non-isomorphic when they result in different computed colourings. In fact, for almost all graph pairs, it suffices to run Colour Refinement on the two graphs in parallel and use the computed colours to decide whether the graphs are isomorphic or not [Babai et al. 1980].

The first main parameter of the algorithm with direct links to other areas is its number of iterations until termination. The number of iterations needed to distinguish two non-isomorphic graphs from each other corresponds to the quantifier depth of a distinguishing formula in the counting logic fragment C^2 (see [Krebs and Verbitsky 2015]). As recently discovered, the maximum number of iterations of the Colour Refinement procedure on graphs of order n is either $n-2$ or $n-1$ [Kiefer and McKay 2020].

However, there are very small examples of non-isomorphic pairs of graphs which the Colour Refinement procedure simply fails to distinguish. To render the expressive power of the algorithm more powerful, i.e. to enable it to distinguish more graphs, one can apply a higher-dimensional variant of it. Instead of colouring the vertices, the k-dimensional WL algorithm (k-WL) colours vertex k-tuples. In fact, 1-WL is Colour Refinement, whereas 2-WL colours arcs, i.e. ordered pairs of vertices. No tight bounds on the number of iterations of higher-dimensional variants of the WL algorithm are known: for all $k \geq 2$, the best known lower bound on the iteration number of k-WL on graphs of order n is $\Omega(n)$ [Fürer 2001]. Concerning upper bounds, only for $k = 2$, significant progress has been made [Kiefer and Schweitzer 2019], the currently best upper bound on the number of iterations of 2-WL being $O(n \log n)$ [Lichter et al. 2019].

Having introduced k-WL, the dimension of the algorithm can be regarded as the second main parameter with close correspondences in other areas. For example, 2-WL is closely connected to the concept of coherent configurations [Evdokimov and Ponomarenko 2000]. In general, the dimension of the WL algorithm that is necessary and sufficient to distinguish a graph from every other graph is exactly one less than the number of variables needed to define the graph in the counting logic C [Cai et al. 1992]. Therefore, it is a measure for the inherent descriptive complexity of the graph.

As shown by Cai, Fürer, and Immerman, to describe graphs on n vertices via C-formulae, $\Omega(n)$ variables are necessary [Cai et al. 1992]. Thus, no fixed dimension of the WL algorithm suffices to decide graph isomorphism in general. Still, it makes perfect sense to study the dimension needed to solve the graph isomorphism problem restricted to certain graph classes. A graph class $\mathcal{G}$ has *WL dimension* at most k if k-WL distinguishes every graph in $\mathcal{G}$ from every non-isomorphic second graph [Grohe 2017]. We say k-WL *identifies* the graph. A finite WL dimension of a graph class yields a polynomial-time isomorphism test for the class (which, however, is mostly of theoretical relevance due to high memory consumption and running time).

For some natural graph classes, their WL dimensions have been determined. For instance, the graph class whose WL dimension is 1 is well-understood [Kiefer et al. 2015; Arvind et al. 2017] and includes, in particular, forests. For other graph classes, their WL dimensions are at least known to be finite. Most notably, every graph class that excludes some graph as a minor has a finite WL dimension [Grohe 2017]. However, for many of the interesting graph classes, no explicit bounds on the WL dimension are known and, typically, the asymptotic ones derived from the proofs of the finiteness cannot be assumed to be anywhere near tight (see, for example, [Grohe 2000; 2012]).

In this column, I consider the class of planar graphs and present the major ingredients of the proof of the following theorem.

THEOREM 1.1. *[Kiefer et al. 2019] The WL dimension of the class of planar graphs is either 2 or 3.*

This bound was obtained in joint work with Ilia Ponomarenko and Pascal Schweitzer and significantly improved over the previously best upper bound of 14 [Grohe 1998; Redies 2014]. The proof presented here uses results from a collaboration with Daniel Neuen to strengthen the insights about the WL algorithm. More precisely, to prove the bound, we reduce the case of general planar graphs to 3-connected ones. For this reduction to work in the given context, we need to show that it can actually be performed by 3-WL. Very surprisingly, it turns out that even 2-WL is able to implicitly decompose a graph into its 3-connected components, which then suffices to conclude the reduction. We finally solve the problem for 3-connected planar graphs by applying a very powerful theorem due to Tutte [Tutte 1963].

Afterwards, I give an overview of how Theorem 1.1 can be generalised to graph classes that are parameterised by their Euler genus. The generalisation is stated in the following result.

THEOREM 1.2. *[Grohe and Kiefer 2019a] The WL dimension of the class of graphs of Euler genus g is at most $4g + 3$.*

In the following two sections, I introduce the WL algorithm and its dimension in more detail. Section 4 treats the reduction from general to 3-connected graphs. Section 5 applies the insights from Section 4 to prove Theorem 1.1. In Section 6, I generalise the results to graphs of arbitrary Euler genus to obtain Theorem 1.2. I conclude with a summary and a discussion of possible future projects in Section 7.

2. THE WL ALGORITHM

This section formally defines the WL algorithm, applied to graphs as input. We always assume graphs to be undirected, i.e. their edges are subsets of size 2 of the vertex set. Still, in the considered edge colourings, we allow the two orientations of an edge to have different colours.

We start off with a presentation of 1-WL, more commonly known as the *Colour Refinement procedure*.

2.1. Colour Refinement

Proceeding in iterations, Colour Refinement computes a so-called *stable colouring* of the vertex set of its input graph. In every iteration, it reassigns to a vertex the multiset of colours of its neighbours from the previous iteration until the vertex partition induced by the colours is not refined anymore. In the following, we use the notation $N(v)$ for the set of neighbours of the vertex v. We generally use the letter χ for colourings computed by the WL algorithm and use λ for arbitrary colourings. We refer to the coloured graph the WL algorithm is applied to as well as its dimension in the index. (However, since these parameters are often unambiguous, we mostly skip in our notation whatever is clear from the context. In particular, since the input colouring λ will always be clear or irrelevant, we omit it in the index.)

Definition 2.1 (*Colour Refinement*). Let $\lambda\colon V(G) \to \mathcal{C}$, where $\mathcal{C}$ is some set of colours, be a vertex colouring of a graph G. The colouring computed by Colour Refinement on input (G, λ) is defined recursively: we set $\chi^0_{G,1} \coloneqq \lambda$, i.e. the initial colouring is λ. For $i \in \mathbb{N}$, the colouring $\chi^i_{G,1}$ computed by Colour Refinement after i iterations on G is defined as

$$\chi^i_{G,1}(v) \coloneqq \big(\chi^{i-1}_{G,1}(v), \{\!\{\chi^{i-1}_{G,1}(w) \,\big|\, w \in N(v)\}\!\}\big).$$

That is, $\chi^i_{G,1}(v)$ consists of the colour of v from the previous iteration as well as the multiset of colours of neighbours of v from the previous iteration. For a colouring $\lambda\colon V(G) \to \mathcal{C}$, denote by $\pi(\lambda)$ the vertex partition induced by λ. It is not difficult to see that the reassignment refines the previous colouring. Therefore, there is a unique minimal integer j such that $\pi(\chi^j_G)$ is as fine as $\pi(\chi^{j+1}_{G,1})$. For this value j, we define the *output* of Colour Refinement on input G to be $\chi_{G,1} \coloneqq \chi^j_{G,1}$.

Colour Refinement can be implemented to run in time $O((m+n)\log n)$, where n is the order of the input graph and m is the number of its edges (see, for example, [Cardon and Crochemore 1982; McKay 1981; Paige and Tarjan 1987]). Figure 1 displays an application of Colour Refinement to a path. Note that the information of being adjacent to a vertex of a "special" colour is propagated to the centre of the path and then the algorithm terminates.

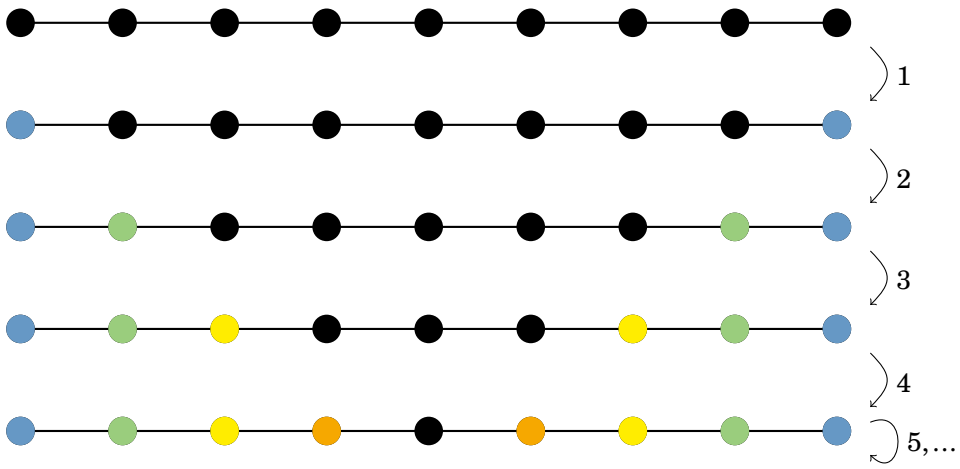

Fig. 1: The iterations of Colour Refinement when applied to an uncoloured path of length 8.

For two graphs G and G', Colour Refinement *distinguishes* G and G' if the multisets of colours in the images of $\chi_{G,1}$ and $\chi_{G',1}$ are distinct. If the graphs are not distinguished by Colour Refinement, we call them *equivalent* with respect to Colour Refinement. The algorithm *identifies* G if it distinguishes G from every other, non-isomorphic graph G'. The definition of the algorithm is isomorphism-invariant, i.e. the output only depends on the isomorphism class of the input graph. Thus, whenever Colour Refinement distinguishes two graphs from each other, they are non-isomorphic.

The simplest example of two graphs that are indistinguishable by 1-WL is formed by a monochromatic hexagon and the disjoint union of two monochromatic triangles, i.e. two 2-regular graphs on six vertices each: every vertex has exactly two neighbours and all vertices have the same colour. However, although the algorithm fails on such basic graphs, it is not difficult to see that 1-WL identifies, for example, every vertex-coloured forest. Much more strongly, on almost all graphs G, the colouring computed by Colour Refinement after two iterations is discrete on $V(G)$, i.e. every vertex obtains a unique colour in the final colouring (which implies graph identification, as is not difficult to see) [Babai et al. 1980].

In the next subsection, we generalise the introduced concepts to vertex tuples instead of single vertices.

2.2. Higher Dimensions of the WL Algorithm

Suppose $k \in \mathbb{N}$. To define k-WL, we need some additional notions. An *edge-coloured graph* (G, λ) is a graph G with a function $\lambda\colon \big\{(u,u) \mid u \in V(G)\big\} \cup \big\{(u,v) \mid \{u,v\} \in E(G)\big\} \to \mathcal{C}$, where $\mathcal{C}$ is some set of colours. For the colouring λ, we assume that the set of colours of loops and the set of colours of other arcs are disjoint, that is, we have

$$\big\{\lambda(u,u) \mid u \in V\big\} \cap \big\{\lambda(u,v) \mid u \neq v, \{u,v\} \in E(G)\big\} = \emptyset.$$

The *atomic type* of a vertex k-tuple $\bar{u} = (u_1, \dots, u_k)$ of a vertex- or edge-coloured graph (G, λ) is the set of all atomic facts satisfied by these vertices. That is, $\bar{u} = (u_1, \dots, u_k)$ and $\bar{v} = (v_1, \dots, v_k)$ have the same atomic type if and only if the mapping $u_i \mapsto v_i$ is an isomorphism from the induced coloured subgraph $G[\{u_1, \dots, u_k\}]$ to the induced coloured subgraph $H[\{v_1, \dots, v_k\}]$.[1] We denote the atomic type of $\bar{u} \coloneqq (u_1, \dots, u_k)$ in (G, λ) by $\operatorname{atp}(G, \lambda, \bar{u})$. Note that, for $k = 1$, the atomic type of a vertex v in a vertex-coloured graph (G, λ) simply consists of $\lambda(v)$. For $k \geq 2$, the atomic type $\operatorname{atp}(G, \lambda, \bar{u})$ can be encoded as a $(k \times k)$-matrix M with

$$M_{ij} = \begin{cases} (0, \lambda\big(u_i, u_i\big)), & \text{if } u_i = u_j \\ (1, \lambda\big(u_i, u_j\big)), & \text{if } (u_i, u_j) \in E(G), u_i \neq u_j \\ (2, \bot), & \text{if } (u_i, u_j) \notin E(G), u_i \neq u_j. \end{cases}$$

We describe the colourings $\chi^i_{G,k}$ computed by k-WL on input a coloured graph (G, λ). (Again, λ will always be unambiguous and we omit it in the index.) The colouring $\chi^0_{G,k}$ assigns to each tuple its atomic type:

$$\chi^0_{G,k}(\bar{u}) \coloneqq \operatorname{atp}(G, \lambda, \bar{u}).$$

For $i \in \mathbb{N}_0$, the colouring $\chi^{i+1}_{G,k}$ computed in the $(i+1)$-st iteration is defined via

$$\chi^{i+1}_{G,k}(\bar{u}) \coloneqq \big(\chi^i_{G,k}(\bar{u}), \mathcal{M}_i(\bar{u})\big),$$

where, for $\bar{u} = (u_1, \dots, u_k)$, the multiset $\mathcal{M}_i(\bar{u})$ is defined as

$$\Big\{\!\!\Big\{ \big(\chi^i_{G,k}(u_1, \dots, u_{k-1}, v), \chi^i_{G,k}(u_1, \dots, v, u_k), \dots, \chi^i_{G,k}(v, u_2, \dots, u_k)\big) \;\Big|\; v \in V(G) \Big\}\!\!\Big\}$$

if $k \geq 2$, and $\mathcal{M}_i(\bar{u}) \coloneqq \Big\{\!\!\Big\{ \chi^i_{G,k}(w) \;\Big|\; w \in N(\bar{u}) \Big\}\!\!\Big\}$ if $k = 1$. Note that, for $k = 1$, the definition coincides with the one for Colour Refinement from Section 2.1.

For $i \in \mathbb{N}_0$, let $\pi^i_{G,k}$ be the partition induced by $\chi^i_{G,k}$ on $(V(G))^k$. Just as for Colour Refinement, $\pi^{i+1}_{G,k}$ refines $\pi^i_{G,k}$ and hence, there is some minimal integer j such that $\pi^j_{G,k} = \pi^{j+1}_{G,k}$. For this value j, we call $\chi^j_{G,k}$ *the stable (k-tuple) colouring* of G and denote it by $\chi_{G,k}$.

For $k > 1$, the algorithm *k-WL* takes as input an edge-coloured graph (G, λ) and computes $\chi_{G,k}$. For $k = 1$, we assume input graphs are only vertex-coloured, see Section 2.1. A bound on the running time of k-WL is $O(n^{k+1} \log n)$, where n is the order of the input graph [Immerman and Lander 1990].

If the graph G or the dimension k is clear from the context, we omit it in the subscripts of χ and π. Distinction, equivalence, and identification of graphs with respect to

[1] For the reader not familiar with this notion, the subgraph of G *induced* by a set of vertices $V' \subseteq V(G)$ is the graph $G[V']$ with $V(G[V']) \coloneqq V'$ and $E(G[V']) \coloneqq E(G) \cap \{\{u,v\} \mid u, v \in V'\}$.

k-WL are defined analogously as for 1-WL. See [Kiefer 2020] for a detailed introduction to k-WL.

2.3. Connections to Logic and Games

Even though the description of the WL algorithm is simple, as soon as its dimension is larger than 2, it becomes difficult to understand how the colourings computed by the algorithm evolve during its execution and which information they precisely encode. Already arguing that two vertex tuples obtain different or equal colours in the stable colouring becomes cumbersome. Luckily, Cai, Fürer, and Immerman found precise correspondences between the algorithm and two other areas from theoretical computer science, namely logics and games, which we briefly discuss in the following [Cai et al. 1992]. Besides these connections, the WL algorithm has links to plenty of other fields (see, for example, [Atserias and Maneva 2013; Grädel et al. 2019; Grohe and Otto 2015]).

Counting Logics. A very fruitful connection of the WL algorithm exists to logics (see also [Grohe 2017]). Denote by C the extension of first-order logic FO by *counting quantifiers* of the form $\exists^{\geq m} x$ with the obvious meaning. C has the same expressive power as FO. Nevertheless, the situation changes when considering the fragments C^k, which consist of all C-formulae with at most k distinct variables (which can, however, be reused): if $m > k$, then $\exists^{\geq m} x$ cannot be expressed in the k-variable fragment of FO.

We write $\varphi(x_1, \ldots, x_\ell)$ to indicate that the free variables of φ are among $x_1, \ldots, x_\ell$. Then for a graph G and vertices $u_1, \ldots, u_\ell \in V(G)$, we write $G \models \varphi(u_1, \ldots, u_\ell)$ to denote that G satisfies φ if, for all i, the variable x_i is interpreted by u_i.

A graph G is *definable* in a logic L if there is an L-sentence iso_G such that for every graph H, it holds that

$$H \models \mathsf{iso}_G \iff G \cong H.$$

As we will see, determining the dimension of the WL algorithm that is necessary and sufficient to identify a particular graph is equivalent to counting how many distinct variables are necessary and sufficient to define the graph in C.

Before stating the precise correspondence between the WL algorithm and C, we treat a second link of the algorithm, which is often helpful when arguing about its power.

Pebble Games. The following is a type of Ehrenfeucht-Fraïssé game that captures the evolution of the colours computed by the WL algorithm when applied to two graphs.

Let $k \in \mathbb{N}$. For graphs G and H with equal numbers of vertices and vertex colourings λ and λ', respectively, we define the *bijective k-pebble game* $\mathrm{BP}_k(G, H)$ as follows (see also [Hella 1996]). The game is played by the players *Spoiler* and *Duplicator*. It proceeds in rounds, each of which is associated with a pair of configurations $(\bar{v}, \bar{w})$ with $\bar{v} \in \big(V(G)\big)^\ell$ and $\bar{w} \in \big(V(H)\big)^\ell$, where $\ell \in \{0, \ldots, k\}$. If not specified otherwise, the initial configuration is a pair of empty tuples.

In the following, we describe one round of the game. Suppose the current configuration is $(\bar{v}, \bar{w}) = \big((v_1, \ldots, v_\ell), (w_1, \ldots, w_\ell)\big)$ with $\ell \in \{0, \ldots, k\}$. Now Spoiler picks an $i \in \{1, \ldots, k\}$. Then Duplicator chooses a bijection $f \colon V(G) \to V(H)$ and Spoiler picks a vertex $v \in V(G)$ and defines $w \coloneqq f(v)$. If $i \leq \ell$, the new configuration of the game is the tuple

$$\big((v_1, \ldots, v_{i-1}, v, v_{i+1}, \ldots, v_\ell), (w_1, \ldots, w_{i-1}, w, w_{i+1}, \ldots, w_\ell)\big).$$

Otherwise, the new configuration is

$$\big((v_1, \ldots, v_\ell, v), (w_1, \ldots, w_\ell, w)\big).$$

Let $\big((v_1,\dots,v_{\ell'}),(w_1,\dots,w_{\ell'})\big)$ denote the new configuration. Recall that $\ell' \leq k$ must hold. Now Spoiler wins the play (after the current round) if there is an $i \in \{1,\dots,\ell'\}$ such that $\lambda(v_i) \neq \lambda'(w_i)$, or there are $i,j \in \{1,\dots,\ell'\}$ such that $v_i = v_j \not\Leftrightarrow w_i = w_j$ or $\{v_i,v_j\} \in E(G) \not\Leftrightarrow \{w_i,w_j\} \in E(H)$. That is, Spoiler wins if the induced ordered subgraphs of G and H are not isomorphic. Otherwise, the game continues with the next round. If there is no configuration of the play such that Spoiler wins, i.e. if the game continues forever, then Duplicator wins.

The interpretation of a configuration $\big((v_1,\dots,v_\ell),(w_1,\dots,w_\ell)\big)$ is that ℓ distinguishable pairs of pebbles are placed on the vertices, the i-th pair being positioned on v_i and w_i.

We say that Spoiler (and Duplicator, respectively) *wins the game* $\mathrm{BP}_k(G,H)$ if Spoiler (and Duplicator, respectively) has a strategy to win the game.

Recalling the definition of the counting logics, we can now phrase the correspondence between the WL algorithm, logics, and games.

THEOREM 2.2 ([CAI ET AL. 1992; IMMERMAN AND LANDER 1990]). *Let $k \in \mathbb{N}_0$. Let G and H be graphs, possibly vertex-coloured, with $|V(G)| = |V(H)|$, and suppose $\bar{u} \coloneqq (u_1,\dots,u_k) \in \big(V(G)\big)^k$ and $\bar{v} \coloneqq (v_1,\dots,v_k) \in \big(V(H)\big)^k$. Then the following are equivalent:*

(1) $\chi_{G,k}(\bar{u}) = \chi_{H,k}(\bar{v})$
(2) $G \models \varphi(\bar{u}) \iff H \models \varphi(\bar{v})$ *holds for all* C^{k+1}*-formulae* $\varphi(x_1,\dots,x_k)$.
(3) Duplicator wins the game $\mathrm{BP}_{k+1}(G,H)$ *with the initial configuration* $(\bar{u},\bar{v})$.

In fact, the connection is even more precise: if G and H are not equivalent with respect to k-WL, then the number of iterations of the algorithm needed to distinguish G and H is exactly one less than the quantifier depth of a distinguishing C^{k+1}-formula and also one less than the number of rounds that Spoiler needs to win against Duplicator in the pebble game $\mathrm{BP}_{k+1}(G,H)$.

3. THE WL DIMENSION

The WL algorithm can be used to detect the non-isomorphism of two given graphs by computing the stable colourings and rejecting if there is a colour c such that the numbers of c-coloured vertex tuples in the two graphs differ. However, even if the stable colourings agree in all colours, the graphs might not be isomorphic. In the light of this phenomenon, Grohe introduced the notion of the WL dimension of a graph as a measure for its inherent structural complexity (see [Grohe 2017, Definition 18.4.3]).

Definition 3.1. Let G be a graph. The *WL dimension* of G is the smallest $k \in \mathbb{N}$ such that k-WL identifies G. Similarly, for a class $\mathcal{G}$ of graphs, its *WL dimension* is the smallest $k \in \mathbb{N}$ such that every graph in $\mathcal{G}$ has WL dimension at most k, and ∞ if no such k exists.

By the correspondence stated in Theorem 2.2, we obtain the following insight, which will prove to be very useful when determining bounds on the WL dimension of a graph class.

COROLLARY 3.2. *A graph has WL dimension at most k if and only if it is definable in* C^{k+1}.

An upper bound of $n-1$ on the WL dimension of a graph with n vertices follows straight from its definition, e.g. via the correspondence to pebble games. Therefore, for a graph class that contains up to isomorphism only finitely many graphs, an upper

bound on its WL dimension can be obtained easily. However, this task can become much more challenging when considering graph classes that contain infinitely many different isomorphism types.

The class of all graphs has WL dimension ∞ (see [Cai et al. 1992]). Nevertheless, the situation may be different when restricting the input to come from a particular subclass of graphs: if one of the input graphs has certain structural properties, these might be exploited by the algorithm when comparing it with other graphs, possibly from outside the class.

Suppose we intend to determine the WL dimension of a graph class that is hereditary, i.e. closed under taking induced subgraphs. Then, as the following lemma states, we can restrict ourselves to the connected graphs in the class. This is easy to see, since 2-WL can detect reachability of one vertex from another.

LEMMA 3.3. *Let $\mathcal{G}$ be a hereditary graph class and suppose $k \geq 2$. If every connected graph in $\mathcal{G}$ has WL dimension at most k, then the WL dimension of $\mathcal{G}$ is at most k.*

For the further reductions we are going to perform when trying to determine the WL dimension of certain graph classes, the following property will become important.

Definition 3.4. Let $\mathcal{H}$ be a set of graphs and let $k \in \mathbb{N}$. We say that k-WL *determines (vertex) orbits on* $\mathcal{H}$ if for every pair of edge-coloured graphs $(G, \lambda), (G', \lambda')$ with $G, G' \in \mathcal{H}$ and all vertices $v \in V(G)$ and $v' \in V(G')$, the following holds: there exists an isomorphism from (G, λ) to (G', λ') mapping v to v' if and only if $\chi_{G,k}(v) = \chi_{G',k}(v')$.

It is not difficult to see that if k-WL determines vertex orbits on $\mathcal{H}$, then it also distinguishes every pair of non-isomorphic graphs in $\mathcal{H}$ from each other.

4. REDUCTION FROM GENERAL TO 3-CONNECTED GRAPHS

In the next section, we show an upper bound on the WL dimension of planar graphs. By Lemma 3.3, once we accept WL dimension at least 2, it suffices to restrict the analysis to connected planar graphs, since the class of planar graphs is hereditary. In general, decomposing a graph into components of higher connectivity is a powerful combinatorial tool when dealing with graphs whose structure as a whole is too complex. For example, the decomposition of a graph into its 2-connected components is isomorphism-invariant and has a tree structure. A similar statement holds for the decomposition of a graph G into its "3-connected components", which are minors and not necessarily subgraphs of G. With respect to the identification of graphs, the canonical decompositions and the tree structure allow to reduce the general problem to the one for 3-connected graphs via an inductive approach.

We intend to follow this path in order to identify general planar graphs. Note that one can expect the problem to become significantly simpler when the input graph can be assumed to be 3-connected, additionally to being planar. Indeed, in that case, by a powerful result due to Whitney, we know that the plane embedding of the graph is combinatorially unambiguous [Whitney 1932]. However, the challenging part about the outlined plan is that, in order to obtain a strong upper bound on the WL dimension of planar graphs, the decomposition must be implicitly performed by a very low dimension of the WL algorithm, since we are not allowed to exceed its power.

In this section, we revisit aspects of graph decompositions into 2- and 3-connected components. We apply the insights in Section 5 in order to distinguish planar graphs with 3-WL.

The results presented in the following two subsections were obtained in collaboration with Ilia Ponomarenko and Pascal Schweitzer. The detailed proofs can be found in [Kiefer et al. 2019]. The contents of Subsection 4.3 are based on joint work with Daniel

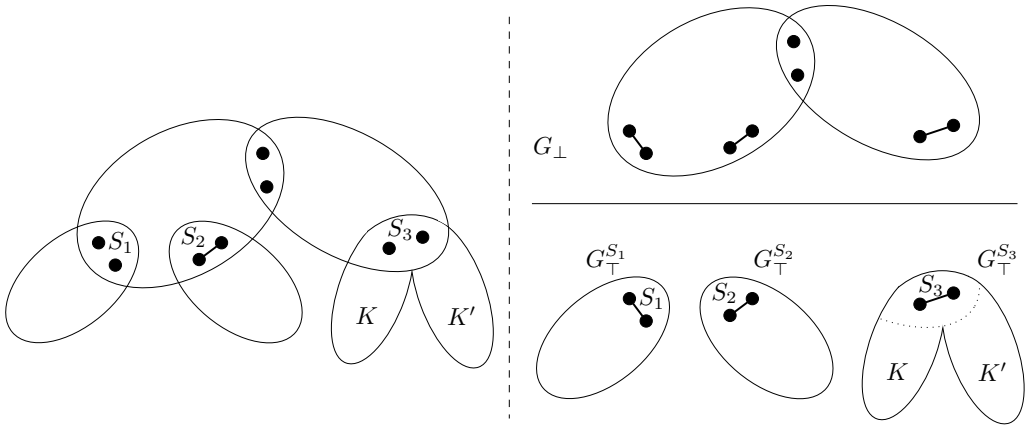

Fig. 2: Schematic illustrations of $G_\perp$ and $G_\top^S$. In the 2-connected graph on the left, the 2-separators are depicted and the respective decomposed graphs are shown on the right.

Neuen. I refer to [Kiefer and Neuen 2019a] and the full version [Kiefer and Neuen 2019b] for proof details.

4.1. Graph Decompositions

For $k \in \mathbb{N}$, a graph G is *k-connected* if G has more than k vertices and for all $X \subseteq V(G)$ with $|X| < k$, the graph $G - X$ is connected. (Here, $G - X \coloneqq G[V(G) \setminus X]$.) A *$k$-separator* of G is a set $S \subseteq V(G)$ with $|S| = k$ for which there are vertices $u, v \in V(G) \setminus S$ that belong to the same connected component of G, but to different connected components of $G - S$. 1-separators are also called *cut vertices*. Define $P(G)$ as the set consisting of the tuples (S, K), where S is a separator of G of minimum cardinality and K is the vertex set of a connected component of $G - S$.

We define a partial order $\leq$ on $P(G)$ by setting

$$(S, K) \leq (S', K') \iff K \subseteq K'.$$

Under the aforementioned assumptions for G, this partial order is well-defined. Let $P_0(G)$ be the set of minimal elements of $P(G)$ with respect to $\leq$. If G is connected but not 3-connected, the elements of $P_0(G)$ correspond to the leaves in a suitable decomposition tree (i.e. the decomposition into 2- or 3-connected components).

In the following, we present a method to remove the vertices appearing in the second component of elements from $P_0(G)$ in a canonical way, which will then let us devise an inductive isomorphism test. In the next two subsections, we show that, surprisingly, already 2-WL implicitly performs this induction, yielding a reduction from general to 3-connected graphs.

For a graph G and a set $S \subseteq V(G)$, let $G_\top^S$ be the graph with vertex set

$$V' \coloneqq S \cup \bigcup_{(S,K) \in P_0(G)} K$$

and edge set $E' \coloneqq E(G[V']) \cup \{\{s, s'\} \mid s, s' \in S \text{ and } s \neq s'\}$, see Figure 2 left and bottom right.

For an edge colouring λ of G, we define an edge colouring $\lambda^S_\top$ for $G^S_\top$ by setting

$$\lambda^S_\top(v_1,v_2) \coloneqq \begin{cases} (0,0) & \text{if } \{v_1,v_2\} \subseteq S \text{ and } \{v_1,v_2\} \notin E(G) \\ (\lambda(v_1,v_2),1) & \text{if } \{v_1,v_2\} \subseteq S \text{ and } \{v_1,v_2\} \in E(G) \\ (\lambda(v_1,v_2),2) & \text{otherwise.} \end{cases}$$

If G is a vertex-coloured graph with vertex colouring λ', in order to obtain a colouring for $G^S_\top$, define an edge colouring λ as $\lambda(v_1,v_2) \coloneqq \lambda'(v_1)$ and let $\lambda^S_\top$ be as above.

For $(S,K) \in P_0(G)$, let $G^{(S,K)}_\top \coloneqq G^S_\top[S \cup K]$. We associate with $G^{(S,K)}_\top$ a colouring $\lambda^{(S,K)}_\top$, the restriction of $\lambda^S_\top$ to pairs (v_1,v_2) with $v_1,v_2 \in S \cup K$.

Given a graph G, let $G_\perp$ (see Figure 2 left and top right) be the graph with vertex set

$$V_\perp \coloneqq V(G) \setminus \left(\bigcup_{(S,K)\in P_0(G)} K \right)$$

and edge set

$$E_\perp \coloneqq E(G[V_\perp]) \cup \big\{ \{s_1,s_2\} \mid \exists (S,K) \in P_0(G) \text{ s.t. } s_1,s_2 \in S, s_1 \neq s_2 \big\}.$$

If G is not 2-connected, then $G_\perp = G[V_\perp]$ holds. If G is not 3-connected and satisfies certain degree conditions (see [Kiefer et al. 2019, Lemma 3]), then $G_\perp$ is a minor of G.

In the following, we restrict ourselves to graphs that are not 3-connected. Given an edge colouring λ of G, define an edge colouring $\lambda_\perp$ of $G_\perp$ as follows. Assume that $v_1, v_2 \in V(G_\perp)$, possibly with $v_1 = v_2$. Let $S \coloneqq \{v_1,v_2\}$.

If S is a 2-separator of G, but $S \notin E(G)$, set

$$\lambda_\perp(v_1,v_2) \coloneqq \left(0, \text{ISOTYPE}\left(\left(G^S_\top, \lambda^S_\top\right)_{(v_1,v_2)} \right)\right).$$

Furthermore, if $v_1 = v_2$ or $\{v_1,v_2\} \in E(G)$, set

$$\lambda_\perp(v_1,v_2) \coloneqq \left(\lambda(v_1,v_2), \text{ISOTYPE}\left(\left(G^S_\top, \lambda^S_\top\right)_{(v_1,v_2)} \right)\right),$$

where the expression $\text{ISOTYPE}((G^S_\top, \lambda^S_\top)_{(v_1,v_2)})$ denotes the isomorphism class of the coloured graph $(G^S_\top, \lambda^S_\top)_{(v_1,v_2)}$ obtained from the coloured graph $(G^S_\top, \lambda^S_\top)$ by individualising the vertices v_1 and v_2, i.e. by assigning to each of them a unique colour (which therefore identifies the vertex).

If not stated otherwise, we implicitly assume that for a graph G with initial colouring λ, the corresponding graph $G_\perp$ is a coloured graph with initial colouring $\lambda_\perp$. The following lemma implies that the described colourings maintain all information that is necessary to determine the isomorphism class of the original graph G. For the proof, see [Kiefer et al. 2019, Lemma 4].

LEMMA 4.1. *For $k \in \{1,2\}$, if G and G' are k-connected graphs that are not $(k+1)$-connected and that are of minimum degree at least $\frac{3k-1}{2}$ with edge colourings λ and λ', respectively, then*

$$(G,\lambda) \cong (G',\lambda') \iff (G_\perp,\lambda_\perp) \cong (G'_\perp,\lambda'_\perp).$$

4.2. Reduction to 2-Connected Graphs

With the insights obtained about decompositions of graphs, we can now state a reduction from general to 2-connected graphs.

THEOREM 4.2. *Let $\mathcal{G}$ be a hereditary graph class. If, for $k \geq 2$, it holds that k-WL distinguishes every two non-isomorphic 2-connected vertex-coloured graphs (G,λ)*

and (G', λ') with $G, G' \in \mathcal{G}$ from each other, then k-WL distinguishes all non-isomorphic vertex-coloured graphs in $\mathcal{G}$.

The following are the basic insights to prove the theorem.

THEOREM 4.3. *Assume $k \geq 2$ and let G and G' be two graphs. Let u and v be vertices from the same 2-connected component of G and let u' and v' be vertices that are not contained in a common 2-connected component of G'. Then $\chi^k_G(u, v) \neq \chi^k_{G'}(u', v')$.*

COROLLARY 4.4. *Let $k \geq 2$ and assume G and G' are connected graphs. Suppose that $w \in V(G)$ and $w' \in V(G')$ are vertices such that $G - \{w\}$ is connected and $G' - \{w'\}$ is disconnected. Then $\chi^k_G(w) \neq \chi^k_{G'}(w')$.*

The proof of the theorem uses the ability of 2-WL to count for every ℓ the walks (i.e. paths with possible vertex repetitions) of length ℓ between two given vertices. Intuitively, in the corresponding game BP_3 with three pebble pairs, we can use two pebble pairs to mark u and u' as well as v and v'. If there is an $\ell \in \mathbb{N}$ such that the numbers of walks from u to v and from u' to v' differ, we can use the third pebble pair to spot this difference. Otherwise, we can use it to mark a cut vertex w' between u' and v' and a vertex $w \in V(G)$ that is not a cut vertex and show that, now counting walks to w and w', respectively, will yield a difference.

To see that the theorem implies the corollary, note that 2-WL can distinguish w and w' because w' has neighbours u' and v' that do not share a 2-connected component, but there are no such neighbours for w.

Knowing that 2-WL distinguishes vertex pairs that share a 2-connected component from other vertex pairs and also assigns special colours to cut vertices, we can now sketch the proof of Theorem 4.2.

PROOF SKETCH FOR THEOREM 4.2. Recall that, by Lemma 3.3, it suffices to consider connected graphs. From Theorem 4.3 and Corollary 4.4, it can be deduced that for connected graphs G and G' which are not 2-connected and vertices $v \in V(G_\perp)$ and $w \in V(G') \setminus V(G'_\perp)$, it holds that $\chi_{G,2}(v) \neq \chi_{G',2}(w)$. This means, informally speaking, that the algorithm detects heights of vertices in the decomposition tree. Moreover, using Theorem 4.3, it also "sees" 2-connected components of vertices in the decomposition tree.

We can then deduce that, for every cut vertex s which occurs in a first component in $P_0(G)$, i.e. that is at the "bottom level" in the decomposition tree, the colour that 2-WL computes for s in G encodes the isomorphism class of the subtree of the decomposition tree rooted at s. This implies that the partition of the vertices and edges of $G_\perp$ induced by the restriction of $\chi_{G,2}$ to pairs of vertices from $V(G_\perp) \subseteq V(G)$ is at least as fine as $\chi_{G_\perp,2}$.

Now the theorem follows by induction. Indeed, suppose the input consists of two vertex-coloured input graphs $(G, \lambda) \ncong (G', \lambda')$ of equal order. If $|V(G)| + |V(G')| = 2$, then the statement is trivial. Otherwise, if at least one of the graphs is 2-connected, the statement follows from the assumptions of the theorem and Corollary 4.4. If neither of G and G' is 2-connected, Lemma 4.1 yields that $(G_\perp, \lambda_\perp) \ncong (G'_\perp, \lambda'_\perp)$. By the induction hypothesis, since $G_\perp$ and $G'_\perp$ are smaller than G and G', 2-WL distinguishes $(G_\perp, \lambda_\perp)$ and $(G'_\perp, \lambda'_\perp)$. Thus, since 2-WL recognises whether a vertex belongs to $V(G_\perp)$ and $V(G'_\perp)$, respectively, and the colourings computed on (G, λ) and (G', λ') refine $\chi_{G_\perp,2}$ and $\chi_{G'_\perp,2}$ on the domains of those, 2-WL also distinguishes (G, λ) and (G', λ'). □

By Theorem 4.2, in particular, a bound on the WL dimension needed to distinguish every pair of 2-connected planar graphs also gives a bound on the dimension needed to distinguish arbitrary planar graphs from each other.

4.3. Reduction to 3-Connected Graphs

In this subsection, the aim is to push the reduction from the previous subsection further and show that, to find bounds on the WL dimension of a (minor-closed) graph class, it suffices to consider the edge-coloured versions of the 3-connected graphs in the class. Put in other words, we will see that 2-WL implicitly computes the decomposition of a graph into its 3-connected components.

The following theorem states the reduction we outline in this subsection. Note that it requires the determination of vertex orbits instead of just distinguishability.

THEOREM 4.5. *Let $\mathcal{G}$ be a minor-closed graph class and assume $k \geq 2$. Suppose k-WL determines vertex orbits on the class of 3-connected graphs in $\mathcal{G}$. Then k-WL distinguishes all non-isomorphic edge-coloured graphs in $\mathcal{G}$.*

Although technically more involved, for $k \geq 3$, the proof of Theorem 4.5 works similarly as the one for Theorem 4.2 and details can be found in [Kiefer et al. 2019].

The proof that the statement in the theorem also holds when $k = 2$ is quite lengthy and described in more detail in [Kiefer and Neuen 2019b]. We outline it in the remainder of this section. Analogously as in the previous subsection, we show that 2-WL implicitly computes the decomposition tree of a graph that is 2-connected but not 3-connected into its 3-connected components. Under certain assumptions, which we can make, these components are actually minors (not necessarily subgraphs) of the input graph. Also, the separators are now pairs of vertices instead of single cut vertices. Therefore, for a statement analogous to the one in Corollary 4.4, we need to show that 2-WL distinguishes 2-separators from other pairs of vertices. This is a surprising fact: recall that, in order to show Theorem 4.3, we used the possibility to mark a cut vertex with a third pebble pair. However, now there are no cut vertices and in order to create one artificially, we have to "block" a vertex contained in a 2-separator, which will then make the second vertex in the pair a cut vertex. But for this, we need an additional pebble pair. Thus, the game is played with 4 pebble pairs and yields only insights about 3-WL. So we cannot take this route.

Instead, in [Kiefer and Neuen 2019a], we perform a structural analysis of all n-vertex graphs. We consider their χ_2-colours and reduce the general case to the case of at most two different χ_2-colours. Finally, we obtain the following insight.

THEOREM 4.6. *Suppose $k \geq 2$. Let G and H be connected graphs. Assume $\{w_1, \dots, w_k\} \subseteq V(G)$ is a k-separator in G. Let $\{v_1, \dots, v_k\} \subseteq V(H)$ and suppose $\chi_{G,k}(w_1, \dots, w_k) = \chi_{H,k}(v_1, \dots, v_k)$. Then $\{v_1, \dots, v_k\}$ forms a k-separator in H.*

Knowing that 2-WL assigns special colours to 2-separators in 2-connected graphs, we can then proceed similarly as outlined in the proof sketch for Theorem 4.2 to show that, in fact, 2-WL implicitly computes the entire decomposition tree of a 2-connected graph that is not 3-connected into its 3-connected components. By induction, this yields Theorem 4.5.

5. THE WL DIMENSION OF PLANAR GRAPHS

This section constitutes the last part of the proof of Theorem 1.1. Again, I refer to the full version for more details on the presented results and their proofs [Kiefer et al. 2019].

By the reduction stated in Theorem 4.5, to show that 3-WL distinguishes all planar graphs from each other, it now suffices to show that the algorithm determines vertex

orbits on the class of 3-connected planar graphs. To this end, we show that, typically, there is a way to individualise two vertices u, v so that an application of 1-WL to the obtained coloured graph $G_{(u,v)}$ yields a discrete colouring. That is, when two suitable vertices are assigned unique vertex colours, then the stable colouring computed by Colour Refinement on the graph will assign a unique colour to *every* vertex. All graphs in which this is not possible will be called exceptions and we treat them separately.

Definition 5.1. We call a graph G an *exception* if G is a 3-connected planar graph in which there are no two vertices v, w in G such that $\chi_{G_{(v,w)},1}$ is a discrete colouring.

The following lemma is a translation of a theorem by Tutte to the context of the WL algorithm.

LEMMA 5.2. *Let G be a 3-connected planar graph and let v_1, v_2, v_3 be vertices of G. If v_1, v_2, v_3 lie on a common face, then $\chi_{G_{(v_1,v_2,v_3)},1}$ is a discrete colouring.*

PROOF. Let v_1, v_2, v_3 be vertices of a common face of G. For $v \in V(G)$, we set $d(v) \coloneqq |N(v)|$. Choose $\mu_0 \colon V(G) \to \mathbb{R}^2$ such that $\mu_0(v_1) = (0,0)$, $\mu_0(v_2) = (1,0)$, $\mu_0(v_3) = (0,1)$, and $\mu_0(v) = (1,1)$ for every $v \in V(G) \setminus \{v_1, v_2, v_3\}$. For $i \in \mathbb{N}_0$, define μ_{i+1} recursively by setting

$$\mu_{i+1}(v) = \begin{cases} \frac{1}{d(v)} \sum_{w \in N(v)} \mu_i(w) & \text{if } v \notin \{v_1, v_2, v_3\}, \\ \mu_i(v) & \text{otherwise.} \end{cases}$$

Then, by Tutte's Spring Embedding Theorem (see [Kobourov 2013, Section 12.3]), the recursion converges to a planar embedding of G [Tutte 1963]. In particular, from a certain i on, the map μ_i is injective.

Recall that we write $\chi^i_{G,k}$ for the colouring computed by the k-dimensional WL algorithm after i iterations on input G. We show that for all $i \in \mathbb{N}_0$ and every two vertices v and v', it holds that

$$\mu_i(v) \neq \mu_i(v') \quad \Rightarrow \quad \chi^i_{G_{(v_1,v_2,v_3)},1}(v) \neq \chi^i_{G_{(v_1,v_2,v_3)},1}(v').$$

We proceed by induction in i. For $i = 0$, the statement holds by the definition of μ_0 and since v_1, v_2, and v_3 have unique colours in $G_{(v_1,v_2,v_3)}$. For the induction step from $i > 0$ to $i+1$, if $\mu_{i+1}(v) \neq \mu_{i+1}(v')$, this implies that $\sum_{w \in N(v)} \mu_i(w) \neq \sum_{w' \in N(v')} \mu_i(w')$. Thus, also the following multiset inequality holds:

$$\{\!\{\mu_i(w) \mid w \in N(v)\}\!\} \neq \{\!\{\mu_i(w') \mid w' \in N(v')\}\!\}.$$

Hence, by induction,

$$\{\!\{\chi^i_{G_{(v_1,v_2,v_3)},1}(w) \mid w \in N(v)\}\!\} \neq \{\!\{\chi^i_{G_{(v_1,v_2,v_3)},1}(w') \mid w' \in N(v')\}\!\}.$$

Therefore, by Definition 2.1, it holds that $\chi^{i+1}_{G_{(v_1,v_2,v_3)},1}(v) \neq \chi^{i+1}_{G_{(v_1,v_2,v_3)},1}(v')$.

Now the fact that from some i on, the map μ_i is injective implies that $\chi^i_{G_{(v_1,v_2,v_3)},1}$ and hence also $\chi_{G_{(v_1,v_2,v_3)},1}$ is a discrete colouring. □

The translation of Tutte's Theorem to the context of the WL algorithm allows us to deduce the following.

COROLLARY 5.3. *4-WL determines vertex orbits on the class of 3-connected planar graphs.*

PROOF. Let G be an edge-coloured 3-connected planar graph and let $n \coloneqq |G|$. Then by Lemma 5.2, there are vertices v_1, v_2, v_3 such that $\chi_{G_{(v_1,v_2,v_3)},1}$ is discrete, since the

additional edge colouring can only refine the stable colouring of the uncoloured graph. This implies that the multiset $C \coloneqq \{\!\{\chi_{G,4}(v_1, v_2, v_3, x) \mid x \in V(G)\}\!\}$ contains n different colours. Let H be a second edge-coloured graph. If H contains vertices v'_1, v'_2, v'_3 such that $\{\!\{\chi_{H,4}(v'_1, v'_2, v'_3, x') \mid x' \in V(H)\}\!\} = C$, then G and H are isomorphic via an isomorphism that maps v_1 to v'_1. Otherwise, the colour $\chi^4_G(v_1, v_2, v_3, v_3)$ is for all $v'_1, v'_2, v'_3 \in V(H)$ different from $\chi^4_H(v'_1, v'_2, v'_3, v'_3)$, which implies that the sets of vertex colours computed by 4-WL in G and H are disjoint. □

In fact, since the graph H is not required to be planar, the proof of the corollary implies that 4-WL also identifies edge-coloured 3-connected planar graphs.

Together with Theorem 4.5, we thus obtain an upper bound of 4 on the WL dimension of the class of planar graphs. To move it down to 3, the strategy is to get by with individualising just two vertices u and v in the statement of the corollary. Whenever such an individualisation enables 1-WL to assign a unique colour also to a third vertex that lies on a common face with u and v, we can apply Lemma 5.2. Assuming this is not possible, the graph is an exception (cf. Definition 5.1). Figure 3 on the following page displays the collection of exceptions, analysed in detail in [Kiefer et al. 2019]. Since the collection is very restricted, a thorough case-by-case analysis shows then that, also on all exceptions, 3-WL determines vertex orbits. Hence, 3-WL distinguishes all planar graphs from each other. In fact, again, the arguments we use in the full version of the proof do not require the second graph to be planar. Therefore, applying Theorem 4.5, we obtain Theorem 1.1.

6. GENERALISATION TO GRAPHS OF ARBITRARY EULER GENUS

Having proved an almost-tight upper bound on the WL dimension of the class of all planar graphs, we now intend to generalise this result to more complex graph classes. More precisely, we consider classes parameterised by their Euler genus. By the classification theorem for surfaces (see [Mohar and Thomassen 2001, Theorem 3.1.3]), up to homeomorphism, which is the topological notion for equivalence, every surface falls into either the family $(\mathbf{S}_k)_{k\geq 0}$ of orientable surfaces or the family $(\mathbf{N}_\ell)_{\ell\geq 1}$ of non-orientable surfaces. The sphere $\mathbf{S}_0$, the torus $\mathbf{S}_1$, and the double torus $\mathbf{S}_2$ are the first three orientable surfaces, and the projective plane $\mathbf{N}_1$ and the Klein bottle $\mathbf{N}_2$ are the first two non-orientable surfaces. We can thus define the *Euler genus* $\mathrm{eg}(\mathbf{S})$ of a surface $\mathbf{S}$ as $2k$ if $\mathbf{S}$ is homeomorphic to the orientable surface $\mathbf{S}_k$, and ℓ if $\mathbf{S}$ is homeomorphic to the non-orientable surface $\mathbf{N}_\ell$. The *Euler genus* of a graph G is the smallest number g such that G is embeddable (that is, can be drawn without edge crossings) in a surface of Euler genus g. See Figure 4 for an example.

The bound presented in the previous section holds for the class of planar graphs, which is the class of graphs embeddable in the sphere $\mathbf{S}_0$. In this section, we establish bounds for graphs embeddable in an arbitrary surface. More formally, we show Theorem 1.2, i.e. that every graph of Euler genus g has WL dimension at most $4g + 3$. This result constitutes the first explicit parameterisation of the WL dimension of a graph by its Euler genus. It was obtained in collaboration with Martin Grohe and the presented contents were published in [Grohe and Kiefer 2019a] (see also the full version [Grohe and Kiefer 2019b]).

To show Theorem 1.2, we use the correspondence from Theorem 2.2 and prove that every graph of Euler genus g is definable in C^{4g+4}. We follow an inductive approach: if the Euler genus is 0, the graph is planar and we can use the results from the previous section. If the Euler genus is higher, the surface will contain a non-contractible curve yielding a so-called *non-contractible cycle* in the embedded graph. Removing the curve results in surfaces of smaller Euler genus. Thus, removing the cycle yields a graph of smaller Euler genus, for which we have a bound on the needed number of variables

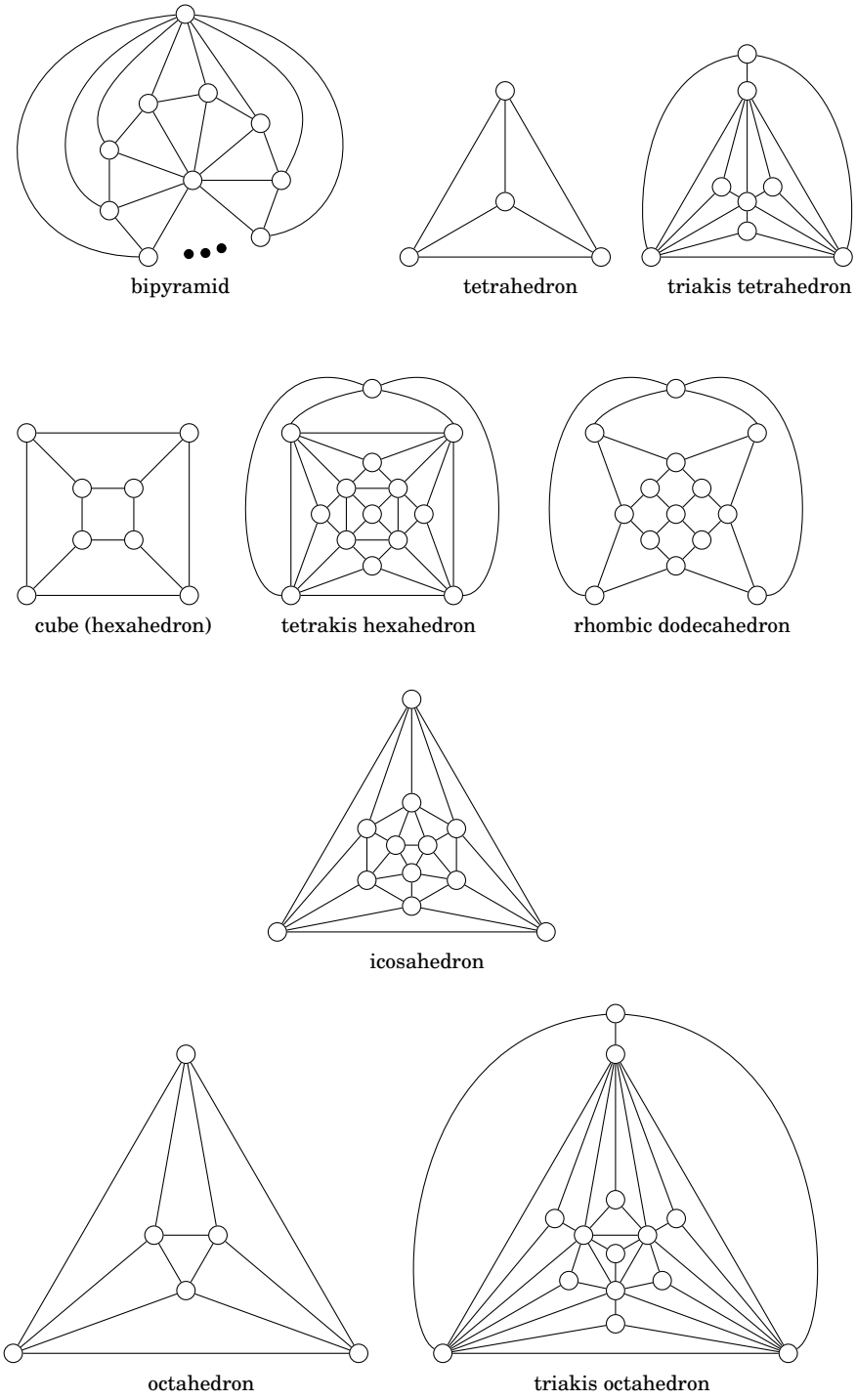

Fig. 3: The 3-connected planar graphs that constitute exceptions.

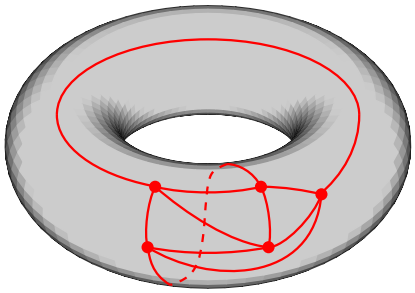

Fig. 4: Embedding of K_5 into the torus $\mathbf{S}_1$.

by induction. In [Grohe and Kiefer 2019a], we show that the removal of the cycle costs 4 variables. Note that, if we start with an orientable surface, the removal of the non-contractible curve will never yield a non-orientable surface. Hence, the Euler genus reduces by at least 2. This yields the following corollary.

COROLLARY 6.1. *The WL dimension of the class of graphs embeddable in an orientable surface of Euler genus g is at most $2g + 3$.*

The remainder of this section is dedicated to an outline of the proof of Theorem 1.2. We start by revisiting some topological concepts. For a more detailed introduction, I refer to [Grohe and Kiefer 2019a] and [Kiefer 2020].

We write topological spaces with bold-face letters. For a topological space $\mathbf{X}$ and a subset $\mathbf{Y} \subseteq \mathbf{X}$, the *boundary* of $\mathbf{Y}$ in $\mathbf{X}$ is the set of all points $x \in \mathbf{X}$ such that every neighbourhood of x has a nonempty intersection with both $\mathbf{Y}$ and $\mathbf{X} \setminus \mathbf{Y}$.

A *simple curve* in a topological space is a homeomorphic image of the real interval $[0, 1]$. Similarly, a *simple closed curve* is a homeomorphic image of $\mathbf{S}_1$. A topological space $\mathbf{X}$ is *arcwise connected* if for any two points $x, y \in \mathbf{X}$, there is a simple curve in $\mathbf{X}$ with endpoints x and y.

A *closed disk* is a homeomorphic image of $\{x \in \mathbb{R}^2 \mid \|x\| \leq 1\}$ equipped with the usual topology, and an *open disk* is a subspace of $\mathbb{R}^2$ homeomorphic to $\mathbb{R}^2$. A simple closed curve g in a surface $\mathbf{S}$ is *contractible* if it is the boundary of a closed disk in $\mathbf{S}$, otherwise g is *non-contractible*. If g is non-contractible, we obtain one or two surfaces of smaller Euler genus by cutting the surface along g and gluing a disk onto each of the obtained holes.

The vertices of a graph *embedded* in a surface $\mathbf{S}$ are points in $\mathbf{S}$, and the edges are simple curves connecting the vertices without edge crossings. Suppose G is a graph embedded in $\mathbf{S}$. A *non-contractible cycle* in G is a cycle C, i.e. a path with identical end vertices, in G such that $\mathbf{C}$ is a non-contractible simple closed curve in $\mathbf{S}$. We denote by $\mathbf{G}$ the subset of $\mathbf{S}$ consisting of all points that are vertices or contained in an edge of G. We say that an abstract graph G' is *embeddable* in a surface $\mathbf{S}$ if it is isomorphic to (the underlying graph of) a graph embedded in $\mathbf{S}$.

A graph G is *polyhedrally embedded* in a surface $\mathbf{S}$ if G is embedded in $\mathbf{S}$, 3-connected, and every non-contractible simple closed curve $\mathbf{g} \subseteq \mathbf{S}$ intersects $\mathbf{G}$ in at least three points. Just like 3-connected graphs embedded in a plane, polyhedrally embedded graphs have nice properties, which we will exploit here.

To proceed via induction, we will use the correspondence from Theorem 2.2: we intend to show that, in order to define graphs of Euler genus g, we need only at most 4 more variables than for defining a graph of smaller Euler genus. This will then yield the factor 4 from Theorem 1.2.

If $g = 0$, the graph is planar and the statement follows from Theorem 1.1.

Suppose $g \geq 1$. Using the reduction to 3-connected graphs stated in Theorem 4.5, in the full version [Grohe and Kiefer 2019b], we show that it suffices to prove the following.

LEMMA 6.2. *Suppose $g \geq 1$ and that there is an $s \geq 4$ such that every graph of Euler genus smaller than g can be defined in C^s. Let G be a coloured graph that is polyhedrally embedded in a surface $\mathbf{S}$ of Euler genus g. Then G is definable in C^{s+3}.*

To show the lemma, we consider two cases as explained in detail below. The goal in the simpler case is to prove that, roughly speaking, fixing just 3 vertices, we can define a subgraph of G whose embedding contains a non-contractible cycle and express how the subgraph is attached to the remainder of the graph. Since the latter is embeddable in a surface of smaller Euler genus and therefore, by the assumptions of the lemma, definable with s variables, we can define the entire graph G with $s + 3$ variables.

The full proof of this informal statement is technically involved. Therefore, we limit ourselves to a presentation of the very basic building blocks and the main structure of the proof. The subgraph which we "cut out" is a so-called *necklace*, a complex structure which even makes an explicit reference to the embedding of the graph. However, since the graph G is just embeddable and not actually embedded, this reference is in its nature highly problematic for the definability of the necklace in C. In order to obtain something definable, we need to base the necklace construction on definable objects as well. It turns out that shortest paths serve very well as these objects.

Definition 6.3. Let G be a graph and $u, u' \in V(G)$. A *shortest path system (sps) from u to u'* is a family $\mathcal{Q}$ of shortest paths in G from u to u' such that every shortest path from u to u' in the subgraph $\bigcup_{Q \in \mathcal{Q}} Q$ is contained in $\mathcal{Q}$.

We let $V(\mathcal{Q}) \coloneqq \bigcup_{Q \in \mathcal{Q}} V(Q)$ and $E(\mathcal{Q}) \coloneqq \bigcup_{Q \in \mathcal{Q}} E(Q)$ and $G(\mathcal{Q}) \coloneqq \big(V(\mathcal{Q}), E(\mathcal{Q})\big) = \bigcup_{Q \in \mathcal{Q}} Q$.

The vertices in $\bigcap_{Q \in \mathcal{Q}} V(Q)$, i.e. the vertices that lie on every shortest path from u to u' in $\mathcal{Q}$, are the *articulation vertices* of $\mathcal{Q}$. An articulation vertex v is *proper* if $v \neq u$ and $v \neq u'$.

For all $u, u' \in V(G)$ such that there is a path from u to u' in G, the *canonical sps from u to u' in G* is the set $\mathcal{Q}^G(u, u')$ of all shortest paths from u to u' in G.

In order to make sure that the final definable object (the necklace) contains a non-contractible cycle, we must restrict our further analysis to special shortest path systems.

Definition 6.4. A *patch* in G is an sps $\mathcal{Q}$ in G such that:

— $\mathcal{Q}$ has no proper articulation vertices.
— There is a closed disk $\mathbf{D} \subseteq \mathbf{S}$ such that $\mathbf{G}(\mathcal{Q}) \subseteq \mathbf{D}$.

As indicated above, patches look highly non-definable in C because they make explicit reference to a specific embedding of G. However, it turns out that, once we are guaranteed that G does not contain a certain type of patch, then this intuition is wrong: in this case, we can define the subgraph contained in a patch. That is, the subgraph becomes independent of the precise embedding of the graph G. We now describe the type of patch whose existence must be excluded for this way of proceeding to work.

We call a subgraph $H \subseteq G$ *simplifying* if every connected component of $G \setminus H \coloneqq G - V(H)$ has Euler genus at most $g - 1$. Otherwise, H is *non-simplifying*.

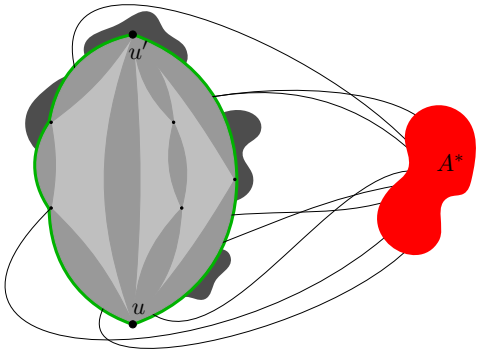

Fig. 5: A patch and some attached connected components. The red component A^* represents the unique non-planar connected component attached to the patch.

A patch $\mathcal{Q}$ is *simplifying* if the graph $G(\mathcal{Q})$ is. Maybe surprisingly, non-simplifying patches can be easier to handle than simplifying ones. The reason is the following result, which, informally stated, allows to define the "outside" of the patch.

LEMMA 6.5 ([GROHE 2017], COROLLARY 15.3.5). *For every non-simplifying subgraph $H \subseteq G$, there is exactly one connected component A^* of $G \setminus H$ such that A^* does not have Euler genus smaller than g, and all other connected components are planar.*

Figure 5 displays a schematic view of a non-simplifying patch with its attached connected components. Note that by Definition 6.4, the connected component A^* must be embedded outside of the embedded patch, due to its own non-planarity. Thus, we can use A^* to mark the outside of the patch and to show that, in fact, the planar subgraph contained inside the embedded patch is definable in our logic. Consequently, non-simplifying patches can be combined to construct more complex definable structures, as we will see.

We are ready to prove Lemma 6.2 via a case distinction.

Case 1: G does not contain any simplifying patches

In this case, we use non-simplifying patches to form the aforementioned necklaces, which we define next.

Definition 6.6. A *necklace* in G is a tuple $\mathcal{B} := (u^0, \mathcal{Q}^0, u^1, \mathcal{Q}^1, u^2, \mathcal{Q}^2)$, where $u^0, u^1, u^2 \in V(G)$ and $\mathcal{Q}^i = \mathcal{Q}^G(u^i, u^{i+1})$ (indices taken modulo 3) is the canonical sps from u^i to u^{i+1}, such that the following conditions are satisfied for every $i \in \{0, 1, 2\}$:

— u^0, u^1, u^2 are pairwise distinct.
— $V(\mathcal{Q}^i) \cap V(\mathcal{Q}^{i+1}) = \{u^{i+1}\}$ (indices modulo 3).
— There is a disk $\mathbf{D}^i \subseteq \mathbf{S}$ such that $\mathbf{G}(\mathcal{Q}^\mathbf{i}) \subseteq \mathbf{D}^i$.

A necklace $\mathcal{B} := (u^0, \mathcal{Q}^1, u^1, \mathcal{Q}^2, u^2, \mathcal{Q}^3)$ is *reducing* if there are paths $Q^i \in \mathcal{Q}^i$ such that $B := Q^1 \cup Q^2 \cup Q^3$ is a non-contractible cycle.

We can think of a reducing necklace as a necklace around a handle or a crosscap of our surface, the beads of the necklace being the disks of the patches. Figure 6 shows a necklace on a torus with articulation vertices u^0, u^0_1, u^1, u^2.

LEMMA 6.7 (NECKLACE LEMMA). *Suppose G is a graph polyhedrally embedded in a surface $\mathbf{S}$ of Euler genus $g \geq 1$ and that G does not contain any simplifying patches. Then G has a reducing necklace.*

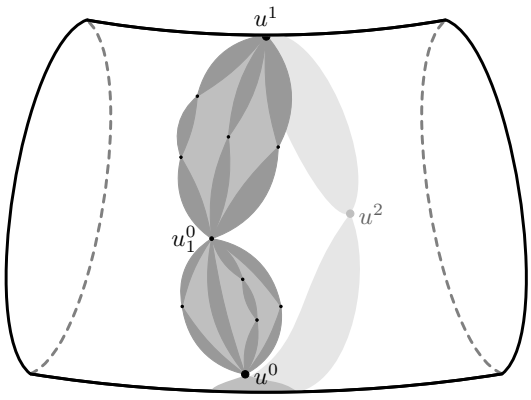

Fig. 6: A necklace on a torus section.

Knowing that G has a reducing necklace, we can proceed as follows. First, we fix a reducing necklace $\mathcal{B} = (u^0, \mathcal{Q}^1, u^1, \mathcal{Q}^2, u^2, \mathcal{Q}^3)$. (To this end, since the $\mathcal{Q}^i$ are canonical sps, it is sufficient to fix u^0, u^1, u^2.) We are going to define a subgraph $\mathrm{Cut}(\mathcal{B})$ that is obtained from G by "cutting through the beads". More precisely, since the necklace is reducing, there is a non-contractible simple closed curve through the necklace, yielding a non-contractible cycle. We can then remove a certain definable supergraph of the cycle to obtain the cut graph $\mathrm{Cut}(\mathcal{B})$. Thus, every connected component of $\mathrm{Cut}(\mathcal{B})$ is embeddable in the surface(s) obtained from cutting S along the non-contractible closed curve and therefore has smaller Euler genus. We can hence define $\mathrm{Cut}(\mathcal{B})$ in C^s using the induction assumption. We then show that the remaining parts of G can be encoded in vertex colours for $\mathrm{Cut}(\mathcal{B})$ such that we can reconstruct G.

To prove that G is definable in the logic C^{s+3}, we need to show that every second graph $\hat{G}$ that satisfies the same C^{s+3}-sentences as G is isomorphic to G. We use the fact that the necklace $\mathcal{B}$ is definable in G by C^4-formulae using the three parameters u^0, u^1, u^2 and define, based on $\mathcal{B}$, the cut graph $\mathrm{Cut}(\mathcal{B})$. Since $\mathrm{Cut}(\mathcal{B})$ has a smaller Euler genus, by the induction assumption, there is a C^s-formula that defines it as an uncoloured graph. We use the additional 3 variables to be able to refer to u^0, u^1, u^2, which is equivalent to individualising these vertices. A thorough analysis then enables us to construct a C^{s+3}-formula $\mathsf{cut\text{-}iso}_G$ that defines the graph $\mathrm{Cut}(\mathcal{B})_{(u_0,u_1,u_2)}$ in G, together with all vertex colours that encode how the necklace $\mathcal{B}$ is attached to the cut graph and therefore allow to reconstruct G.

The formulae for $\mathcal{B}$ and $\mathrm{Cut}(\mathcal{B})$ identify some objects $\hat{\mathcal{B}}$ and $\mathrm{Cut}(\hat{\mathcal{B}})$ in $\hat{G}$. We call those objects a *pseudo-necklace* and a *pseudo-cut graph*. Using $\mathsf{cut\text{-}iso}_G$, we know that the coloured versions of $\mathrm{Cut}(\mathcal{B})$ and $\mathrm{Cut}(\hat{\mathcal{B}})$ are isomorphic. Since we can reconstruct G and $\hat{G}$ from the respective (pseudo-)cut graphs, we conclude that G and $\hat{G}$ are isomorphic.

Case 2: G contains a simplifying patch

At first sight, this case sounds simpler than the first one: we could just remove the simplifying patch instead of a complicated necklace, and the remaining pieces have smaller Euler genus and thus can be identified in the logic C^s. However, we need to make sure not to lose information about the way the pieces are attached to each other. That is, we need to guarantee that we can reconstruct the original graph G. The problem is that simplifying patches have a much more complicated structure than non-simplifying patches. For example, we cannot define the inside of a simplifying patch

in the same way as we did for non-simplifying patches, since there is not necessarily a non-planar connected component that marks the "outside region". Therefore, the idea of the proof is to remove the canonical sps and some interior parts of the corresponding patch, which actually turn out to be definable.

More precisely, we fix two vertices u and u' such that $\mathcal{Q} \coloneqq \mathcal{Q}^G(u, u')$ is a minimal simplifying canonical patch in G, that is, a simplifying patch all of whose proper subpatches are non-simplifying. We extend $\mathcal{Q}$ by the interior graphs of all proper subpatches and obtain a graph J, which is embedded in the disk of $\mathcal{Q}$ and therefore planar.

Now we distinguish between two cases. If $J - \{u, u'\}$ is connected, the patch $\mathcal{Q}$ behaves almost like a non-simplifying patch, and we can argue similarly as in Case 1. If $J - \{u, u'\}$ is disconnected, the patch $\mathcal{Q}$ can be split into several so-called *fibres*. The subgraph J contained in the fibres is definable after fixing one additional particular vertex. In fact, the subgraph contained in every single fibre is definable. We exploit this in order to encode in the boundary vertices of the fibres the way in which the remainder of the graph is attached to them, similarly as in Case 1, but due to the possibly complex structure of J a lot more involved.

7. OUTLOOK

After an introduction to the central concepts of the WL algorithm, this column has given an overview of recent results concerning improved bounds on the WL dimension of the class of planar graphs and, more generally, the classes of graphs parameterised by a bound on their Euler genus. More precisely, we showed that the WL dimension of the class of planar graphs is either 2 or 3 (see Theorem 1.1) and provided a linear parametrisation with good constants of the WL dimension of a graph by its Euler genus (see Theorem 1.2). To obtain these results, we used insights about the ability of 2-WL, 3-WL, and higher dimensions of the WL algorithm to detect substructures in graphs. For example, 2-WL can detect 2-separators in graphs, which implies that it can implicitly compute the decomposition of a graph into its 3-connected components. Also, 3-WL identifies all 3-connected planar graphs and even determines vertex orbits on them.

A goal for future work would be to determine the precise WL dimension of the class of planar graphs. Note that, by Theorem 4.5, it would suffice to show that 2-WL determines vertex orbits on all 3-connected planar graphs and identifies them. Even without identification, it would be major progress to know that 2-WL distinguishes all pairs of 3-connected planar graphs. However, it remains widely open to find, similarly as the one in Figure 3, a useful characterisation of the graphs on which individualising a single vertex and applying Colour Refinement afterwards does not yield a discrete vertex colouring. The collection of such graphs will be much bigger than the set of exceptions depicted in Figure 3 and it is unclear how the new exceptions could be identified by 2-WL.

To show that the WL dimension of the class of planar graphs is 3, one would have to find pairs of non-isomorphic planar graphs that are equivalent with respect to 2-WL. The classical construction for such difficult graphs is the one described by Cai, Fürer, and Immerman in [Cai et al. 1992]. However, already for very low vertex degrees, the construction yields non-planar graphs and can therefore only be of restricted use in the given context.

If 2-WL turns out to identify all planar graphs, the additive constant 4 in the bound from Theorem 1.2 can be replaced with a 3, since C^3 corresponds to 2-WL. Also, by refining the arguments in [Grohe and Kiefer 2019b], it might be possible to bring the constant factor in the statement of the theorem down to 3 or 2. However, without

substantial new ideas, it seems impossible to obtain a better bound than $2g + 3$ on the WL dimension of graphs of Euler genus g.

Concerning lower bounds on the WL dimension in terms of the Euler genus, with the construction described in [Cai et al. 1992], it is not difficult to prove a bound of $\varepsilon \cdot g$, albeit with a small constant factor ε. It remains open how to narrow the gap between this lower and the explicit upper bound presented in this column.

Instead of the Euler genus, one can also find parametrisations of the WL dimension by other graph invariants. By the graph minor theory due to Robertson and Seymour, for every surface, the graphs embeddable in the surface can be completely characterised via a certain list of excluded minors [Robertson and Seymour 1990]. For example, by a version of Kuratowski's Theorem (see [Kuratowski 1930]) due to Wagner, a graph is planar if and only if it excludes both the complete graph K_5 and the complete bipartite graph $K_{3,3}$ as minors [Wagner 1937]. Moreover, in [Archdeacon 1981], the excluded minors that characterise the graphs embeddable in the projective plane are described. Unfortunately, explicit excluded-minor characterisations for more complex surfaces are not known. Still, a natural and interesting target for further study of the WL dimension of a graph class would be the one characterised by the excluded minor K_ℓ. The WL dimension of this class is bounded [Grohe 2017], but so far, it is not even known to be exponentially bounded in ℓ.

REFERENCES

Dan Archdeacon. 1981. A Kuratowski Theorem for the Projective Plane. *Journal of Graph Theory* 5, 3 (1981), 243–246. https://doi.org/10.1002/jgt.3190050305

Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. 2017. Graph Isomorphism, Color Refinement, and Compactness. *Computational Complexity* 26, 3 (2017), 627–685. https://doi.org/10.1007/s00037-016-0147-6

Albert Atserias and Elitza N. Maneva. 2013. Sherali-Adams Relaxations and Indistinguishability in Counting Logics. *SIAM J. Comput.* 42, 1 (2013), 112–137. https://doi.org/10.1137/120867834

László Babai. 2016. Graph Isomorphism in Quasipolynomial Time. In *Proceedings of the Fourty-Eighth Annual ACM Symposium on Theory of Computing*. 684–697. https://doi.org/10.1145/2897518.2897542

László Babai, Paul Erdős, and Stanley M. Selkow. 1980. Random Graph Isomorphism. *SIAM J. Comput.* 9, 3 (1980), 628–635. https://doi.org/10.1137/0209047

Jin-Yi Cai, Martin Fürer, and Neil Immerman. 1992. An Optimal Lower Bound on the Number of Variables for Graph Identification. *Combinatorica* 12 (1992), 389–410. https://doi.org/10.1007/BF01305232

Alain Cardon and Maxime Crochemore. 1982. Partitioning a Graph in $O(|A| \log |A|)$. *Theoretical Computer Science* 19 (1982), 85–98. https://doi.org/10.1016/0304-3975(82)90016-0

Sergei Evdokimov and Ilia N. Ponomarenko. 2000. Separability Number and Schurity Number of Coherent Configurations. *The Electronic Journal of Combinatorics* 7 (2000). https://doi.org/10.37236/1509

Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. 2020. Identifiability of Graphs with Small Color Classes by the Weisfeiler-Leman Algorithm. In *STACS (LIPIcs)*, Vol. 154. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 43:1–43:18. https://doi.org/10.4230/LIPIcs.STACS.2020.43

Martin Fürer. 2001. Weisfeiler-Lehman Refinement Requires at Least a Linear Number of Iterations. In *Proceedings of the Twenty-Eighth International Colloquium, Automata, Languages and Programming*. 322–333. https://doi.org/10.1007/3-540-48224-5_27

Erich Grädel, Martin Grohe, Benedikt Pago, and Wied Pakusa. 2019. A Finite-Model-Theoretic View on Propositional Proof Complexity. *Logical Methods in Computer Science* 15, 1 (2019). https://doi.org/10.23638/LMCS-15(1:4)2019

Martin Grohe. 1998. Fixed-point Logics on Planar Graphs. In *Proceedings of the Thirteenth IEEE Symposium on Logic in Computer Science*. 6–15. https://doi.org/10.1109/LICS.1998.705639

Martin Grohe. 2000. Isomorphism Testing for Embeddable Graphs through Definability. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*. 63–72. https://doi.org/10.1145/335305.335313

Martin Grohe. 2012. Fixed-point Definability and Polynomial Time on Graphs with Excluded Minors. *J. ACM* 59, 5 (2012), 27:1–27:64. https://doi.org/10.1145/2371656.2371662

Martin Grohe. 2017. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Lecture Notes in Logic, Vol. 47. Cambridge University Press. https://doi.org/10.1017/9781139028868

Martin Grohe and Sandra Kiefer. 2019a. A Linear Upper Bound on the Weisfeiler-Leman Dimension of Graphs of Bounded Genus. In *Proceedings of the Fourty-Sixth International Colloquium on Automata, Languages, and Programming*. 117:1–117:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.117

Martin Grohe and Sandra Kiefer. 2019b. A Linear Upper Bound on the Weisfeiler-Leman Dimension of Graphs of Bounded Genus. *Computing Research Repository* (2019). http://arxiv.org/abs/1904.07216

Martin Grohe and Martin Otto. 2015. Pebble Games and Linear Equations. *Journal of Symbolic Logic* 80, 3 (2015), 797–844. https://doi.org/10.1017/jsl.2015.28

Martin Grohe, Pascal Schweitzer, and Daniel Wiebking. 2020. Deep Weisfeiler Leman. *Computing Research Repository* (2020). http://arxiv.org/abs/2003.10935

Lauri Hella. 1996. Logical Hierarchies in PTIME. *Information and Computation* 129, 1 (1996), 1–19. https://doi.org/10.1006/inco.1996.0070

Neil Immerman and Eric Lander. 1990. Describing Graphs: A First-order Approach to Graph Canonization. In *Complexity theory retrospective*, Alan L. Selman (Ed.). Springer, 59–81. https://doi.org/10.1007/978-1-4612-4478-3_5

Sandra Kiefer. 2020. *Power and Limits of the Weisfeiler-Leman Algorithm*. Ph.D. Dissertation. RWTH Aachen University, Aachen. https://doi.org/10.18154/RWTH-2020-03508

Sandra Kiefer and Brendan D. McKay. 2020. The Iteration Number of Colour Refinement. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020) (Leibniz International Proceedings in Informatics (LIPIcs))*, Artur Czumaj, Anuj Dawar, and Emanuela Merelli (Eds.), Vol. 168. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 73:1–73:19. https://drops.dagstuhl.de/opus/volltexte/2020/12480

Sandra Kiefer and Daniel Neuen. 2019a. The Power of the Weisfeiler-Leman Algorithm to Decompose Graphs. In *Proceedings of the Fourty-Fourth International Symposium on Mathematical Foundations of Computer Science*. 45:1–45:15. https://doi.org/10.4230/LIPIcs.MFCS.2019.45

Sandra Kiefer and Daniel Neuen. 2019b. The Power of the Weisfeiler-Leman Algorithm to Decompose Graphs. *Computing Research Repository* (2019). http://arxiv.org/abs/1908.05268

Sandra Kiefer, Ilia N. Ponomarenko, and Pascal Schweitzer. 2019. The Weisfeiler-Leman Dimension of Planar Graphs Is at Most 3. *Journal of the ACM* 66, 6 (2019), 44:1–44:31. https://doi.org/10.1145/3333003

Sandra Kiefer and Pascal Schweitzer. 2019. Upper Bounds on the Quantifier Depth for Graph Differentiation in First-Order Logic. *Logical Methods in Computer Science* 15, 2 (2019). https://doi.org/10.23638/LMCS-15(2:19)2019

Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. 2015. Graphs Identified by Logics with Counting. In *Proceedings of the Fortieth International Symposium on Mathematical Foundations of Computer Science*, Vol. 9234. Springer, 319–330. https://doi.org/10.1007/978-3-662-48057-1_25

Stephen G. Kobourov. 2013. Force-Directed Drawing Algorithms. In *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, 383–408. https://doi.org/10.1201/b15385

Andreas Krebs and Oleg Verbitsky. 2015. Universal Covers, Color Refinement, and Two-Variable Counting Logic: Lower Bounds for the Depth. In *Proceedings of the Thirtieth Annual ACM/IEEE Symposium on Logic in Computer Science*. 689–700. https://doi.org/10.1109/LICS.2015.69

Casimir Kuratowski. 1930. Sur le Problème des Courbes Gauches en Topologie. *Fundamenta Mathematicae* 15, 1 (1930), 271–283. http://eudml.org/doc/212352

Moritz Lichter, Ilia N. Ponomarenko, and Pascal Schweitzer. 2019. Walk Refinement, Walk Logic, and the Iteration Number of the Weisfeiler-Leman Algorithm. In *Proceedings of the Thirty-Fourth Annual ACM/IEEE Symposium on Logic in Computer Science*. 1–13. https://doi.org/10.1109/LICS.2019.8785694

Brendan D. McKay. 1981. Practical Graph Isomorphism. *Congressus Numerantium* 30 (1981), 45–87.

Bojan Mohar and Carsten Thomassen. 2001. *Graphs on Surfaces*. Johns Hopkins University Press.

Harry L. Morgan. 1965. The Generation of a Unique Machine Description for Chemical Structures – A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation* 5, 2 (1965), 107–113. https://doi.org/10.1021/c160017a018

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan E. Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v33i01.33014602

Robert Paige and Robert E. Tarjan. 1987. Three Partition Refinement Algorithms. *SIAM J. Comput.* 16, 6 (1987), 973–989. https://doi.org/10.1137/0216062

Joachim Redies. 2014. *Defining PTIME Problems on Planar Graphs with Few Variables*. Master's thesis. RWTH Aachen University.

Neil Robertson and Paul D. Seymour. 1990. Graph Minors. IX. Disjoint Crossed Paths. *Journal of Combinatorial Theory, Series B* 49, 1 (1990), 40–77. https://doi.org/10.1016/0095-8956(90)90063-6

William T. Tutte. 1963. How to Draw a Graph. *Proceedings of the London Mathematical Society* 13 (1963), 743–767. https://doi.org/10.1112/plms/s3-13.1.743

Klaus Wagner. 1937. Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.* 114, 1 (1937), 570–590. https://doi.org/10.1007/BF01594196

Hassler Whitney. 1932. Congruent Graphs and the Connectivity of Graphs. *American Journal of Mathematics* 54 (1932), 150–168. https://doi.org/10.2307/2371086