Local Problems on Trees from the Perspectives of Distributed Algorithms, Finitary Factors, and Descriptive Combinatorics

Sebastian Brandt
CISPA Helmholtz Center for Information Security
brandt@cispa.de

Yi-Jun Chang*
National University of Singapore
cyijun@nus.edu.sg

Jan Grebík†
University of Warwick
Jan.Grebik@warwick.ac.uk

Christoph Grunau‡
ETH Zürich
cgrunau@inf.ethz.ch

Václav Rozhoň‡
ETH Zürich
rozhonv@ethz.ch

Zoltán Vidnyánszky§
California Institute of Technology
vidnyanz@caltech.edu

Abstract

We study connections between three different fields: distributed local algorithms, finitary factors of iid processes, and descriptive combinatorics. We focus on two central questions: Can we apply techniques from one of the areas to obtain results in another? Can we show that complexity classes coming from different areas contain precisely the same problems? We give an affirmative answer to both questions in the context of local problems on regular trees:

1. We extend the Borel determinacy technique of Marks [Marks – J. Am. Math. Soc. 2016] coming from descriptive combinatorics and adapt it to the area of distributed computing, thereby obtaining a more generally applicable lower bound technique in descriptive combinatorics and an entirely new lower bound technique for distributed algorithms. Using our new technique, we prove deterministic distributed $\Omega(\log n)$-round lower bounds for problems from a natural class of homomorphism problems. Interestingly, these lower bounds seem beyond the current reach of the powerful round elimination technique [Brandt – PODC 2019] responsible for all substantial locality lower bounds of the last years. Our key technical ingredient is a novel *ID graph* technique that we expect to be of independent interest; in fact, it has already played an important role in a new lower bound for the Lovász local lemma in the Local Computation Algorithms model from sequential computing [Brandt, Grunau, Rozhoň – PODC 2021].
2. We prove that a local problem admits a Baire measurable coloring if and only if it admits a local algorithm with local complexity $O(\log n)$, extending the classification of Baire measurable colorings of Bernshteyn [Bernshteyn – personal communication]. A key ingredient of the proof is a new and simple characterization of local problems that can be solved in $O(\log n)$ rounds. We complement this result by showing separations between complexity classes from distributed computing, finitary factors, and descriptive combinatorics. Most notably, the class of problems that allow a distributed algorithm with sublogarithmic randomized local complexity is incomparable with the class of problems with a Borel solution.

*Supported by Dr. Max Rössler, by the Walter Haefner Foundation, and by the ETH Zürich Foundation.
†Supported by Leverhulme Research Project Grant RPG-2018-424.
‡Supported by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 853109).
§Partially supported by the National Research, Development and Innovation Office – NKFIH, grants no. 113047, no. 129211 and FWF Grant M2779.

We hope that our treatment will help to view all three perspectives as part of a common theory of locality, in which we follow the insightful paper of [Bernshteyn – arXiv 2004.04905].

Contents

1 Introduction

In this work, we study local problems on regular trees from three different perspectives.

First, we consider the perspective of distributed algorithms. In distributed computing, the studied setup is a network of computers where each computer can only communicate with its neighbors. Roughly speaking, the question of interest in this area is which problems can be solved with only a few rounds of communication in the underlying network.

Second, we consider the perspective of (finitary) factors of iid processes. In probability, random processes model systems that appear to vary in a random manner. These include Bernoulli processes, Random walks etc. A particular, well-studied, example is the Ising model.

Third, we investigate the perspective of descriptive combinatorics. The goal of this area is to understand which constructions on infinite graphs can be performed without using the so-called axiom of choice.

Although many of the questions of interest asked in these three areas are quite similar to each other, no systematic connections were known until an insightful paper of Bernshteyn [20] who showed that results from distributed computing can automatically imply results in descriptive combinatorics. In this work, we show that the connections between the three areas run much deeper than previously known, both in terms of techniques and in terms of complexity classes. In fact, our work suggests that it is quite useful to consider all three perspectives as part of a common theory, and we will attempt to present our results accordingly. We refer the reader to Figure 1 for a partial overview of the rich connections between the three perspectives, some of which are proven in this paper.

In this work, we focus on the case where the graph under consideration is a regular tree. Despite its simplistic appearance, regular trees play an important role in each of the three areas, as we will substantiate at the end of this section. To already provide an example, in the area of distributed algorithms, the majority of known locality lower bounds is achieved on regular trees. Moreover, when regarding lower bounds, the property that they already apply on regular trees actually *strengthens* the result—a fact that is quite relevant for our work as our main contribution regarding the transfer of techniques between the areas is a new lower bound technique in the area of distributed computation that is an adaptation and generalization of a technique from descriptive combinatorics. Regarding our results about the relations between complexity classes from the three areas, we note that such connections are also studied in the context of paths and grids in other recent papers [61, 62].

In the remainder of this section, we give a high-level overview of the three areas that we study. The purpose of these overviews is to provide the reader with a comprehensive picture of the studied settings that can also serve as a starting point for delving deeper into selected topics in those areas—in order to follow our paper, it is not necessary to obtain a detailed understanding of the results and connections presented in the overviews. Necessary technical details will be provided in Section 3. Moreover, in Section 2, we present our contributions in detail.

Distributed Computing The definition of the LOCAL model of distributed computing by Linial [81] was motivated by the desire to understand distributed algorithms in huge networks. As an example, consider a huge network of wifi routers. Let us think of two routers as connected by an edge if they are close enough to exchange messages. It is desirable that such close-by routers communicate with user devices on different channels to avoid interference. In graph-theoretic language, we want to properly color the underlying network. Even if we are allowed a color palette with $\Delta + 1$ colors where Δ denotes the maximum degree of the graph (which would admit a simple greedy algorithm in a sequential setting), the problem remains highly interesting in the distributed

setting, as, ideally, each vertex decides on its output color after only a few rounds of communication with its neighbors, which does not allow for a simple greedy solution.

The LOCAL model of distributed computing formalizes this setup: we have a large network, where each vertex knows the network's size, n, and perhaps some other parameters like the maximum degree Δ. In the case of randomized algorithms, each vertex has access to a private random bit string, while in the case of deterministic algorithms, each vertex is equipped with a unique identifier from a range polynomial in the size n of the network. In one round, each vertex can exchange any message with its neighbors and can perform an arbitrary computation. The goal is to find a solution to a given problem in as few communication rounds as possible. As the allowed message size is unbounded, a t-round LOCAL algorithm can be equivalently described as a function that maps t-hop neighborhoods to outputs—the output of a vertex is then simply the output its t-hop neighborhood mapped to by this function. An algorithm is correct if and only if the collection of outputs at all vertices constitutes a correct solution to the problem.

There is a rich theory of distributed algorithms and the local complexity of many problems is understood. The case of trees is a highlight of the theory: it is known that any local problem (a class of natural problems we will define later) belongs to one of only very few complexity classes. More precisely, for any local problem, its randomized local complexity is either $O(1), \Theta(\log^* n), \Theta(\log \log n), \Theta(\log n)$, or $\Theta(n^{1/k})$ for some $k \in \mathbb{N}$. Moreover, the deterministic complexity is always the same as the randomized one, except for the case $\Theta(\log \log n)$, for which the corresponding deterministic complexity is $\Theta(\log n)$ (see Figure 1).

(Finitary) Factors of iid Processes and Uniform Algorithms In recent years, *factors of iid (fiid) processes* on trees attracted a lot of attention in combinatorics, probability, ergodic theory and statistical physics [1, 17, 3, 5, 2, 4, 6, 7, 25, 24, 37, 43, 52, 53, 63, 64, 67, 69, 70, 71, 73, 79, 96, 97, 82, 91, 98, 104, 105]. Intuitively, factors of iid processes are randomized algorithms on, e.g., infinite Δ-regular trees, where each vertex outputs a solution to a problem after it explores random strings on vertices of the whole tree. As an example, consider the *perfect matching* problem. An easy parity argument shows that perfect matching cannot be solved by any local randomized algorithm on finite trees. However, if we allow a small fraction of vertices not to be matched, then, by a result of Nguyen and Onak [94] (see also [47]), there is a constant-round randomized algorithm that produces such a matching on high-girth graphs (where the constant depends on the fraction of unmatched vertices that we allow). This result can also be deduced from a result of Lyons and Nazarov [83], who showed that perfect matching can be described as a factor of iid process on an infinite Δ-regular tree. The high-level idea behind this connection is that high-girth graphs approximate the infinite Δ-regular tree and constant-round local algorithms approximate factors of iid processes. This correspondence is formalized in the notion of *Benjamini-Schramm* or *local-global* convergence [18, 65]. We note that getting only "approximate" solutions, that is, solutions where a small fraction of vertices does not have to satisfy the constraints of a given problem, is intrinsic in this correspondence. Regardless, there are many techniques, such as entropy inequality [2] or correlation decay [4], and particular results such as the aforementioned perfect matching problem [83] that provide lower and upper bounds, respectively, in our setting as well. We refer the reader to [82, 6] for a comprehensive summary of the field.

In this paper, we mostly consider a stronger condition than fiid, namely so-called *finitary* factors of iid (ffiid) processes that are studied in the same context as fiid [68, 72, 102]. Perhaps surprisingly, the notion of ffiid is identical to the notion of so-called uniform distributed randomized algorithms [78, 62] that we now describe. We define a uniform local algorithm as a randomized local algorithm that does not know the size of the graph n – this enables us to run such an algorithm on infinite

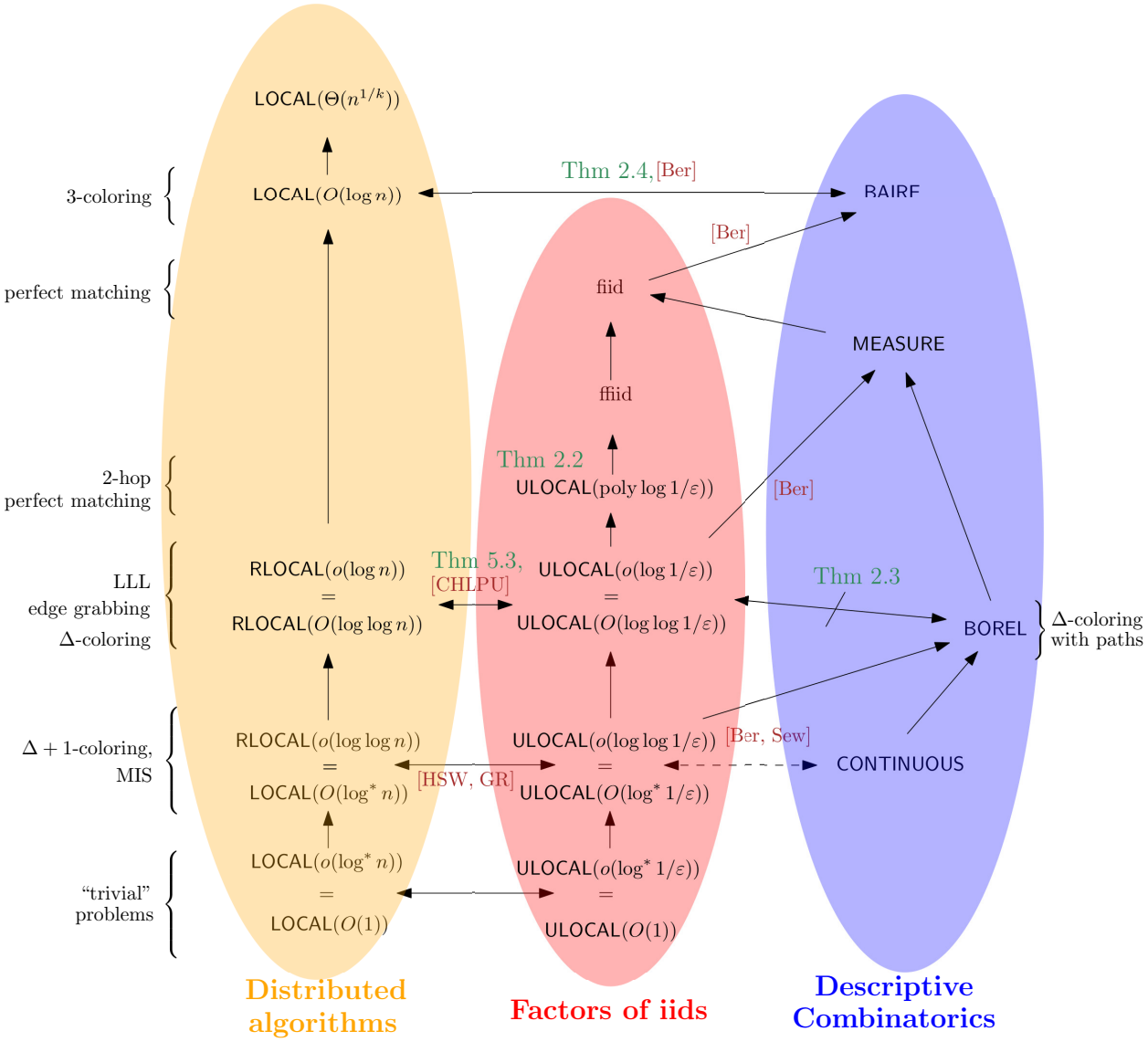

Figure 1: Complexity classes on regular trees considered in the three areas of distributed computing, factors of iid processes/uniform algorithms, and descriptive combinatorics. The left part shows complexity classes of distributed computing. We use the shorthand LOCAL if it does not matter whether we talk about the deterministic or randomized complexity. These two notions differ only for the class of problems of randomized local complexity $O(\log \log n)$, which have deterministic complexity $O(\log n)$.

The uniform complexity classes of sublogarithmic complexity are in correspondence to appropriate classes in the randomized local complexity model, as proven in Section 5. On the other hand, the class fiid is very similar to the class MEASURE from descriptive combinatorics. The equivalence of the class CONTINUOUS and $\mathsf{LOCAL}(O(\log^* n)) = \mathsf{ULOCAL}(O(\log^* 1/\varepsilon))$ is marked with a dashed arrow as it was proven in case the tree is generated by a group action (think of the tree being equipped with an additional Δ-edge coloring). The inclusion $\mathsf{LOCAL}(O(\log^* n)) \subseteq \mathsf{BOREL}$ however clearly holds also in our setting. The class BOREL is incomparable with $\mathsf{RLOCAL}(O(\log \log n))$, as proven in Section 5.

graphs, where there is no n. More precisely, we require that each vertex eventually outputs a solution that is compatible with the output in its neighborhood, but the time until the vertex finishes is a potentially unbounded random variable. As in the case of classical randomized algorithms, we can now measure the *uniform complexity* of a uniform local algorithm (known as the tail decay of ffiid [72]). The uniform complexity of an algorithm is defined as the function $t(\varepsilon)$ such that the probability that the algorithm run on a specific vertex needs to see outside its $t(\varepsilon)$-hop neighborhood is at most ε. As in the case of classical local complexity, there is a whole hierarchy of possible uniform complexities (see Figure 1).

We remark that uniform distributed local algorithms can be regarded as Las Vegas algorithms. The output will always be correct; there is however no fixed guarantee at what point all vertices have computed their final output. On the other hand, a randomized distributed local algorithm can be viewed as a Monte Carlo algorithm as it needs to produce an output after a fixed number of rounds, though the produced output might be incorrect.

Descriptive Combinatorics The Banach-Tarski paradox states that a three-dimensional ball of unit volume can be decomposed into finitely many pieces that can be moved by isometries (distance preserving transformations such as rotations and translations) to form two three-dimensional balls each of them with unit volume(!). The graph theoretic problem lurking behind this paradox is the following: fix finitely many isometries of $\mathbb{R}^3$ and then consider a graph where x and y are connected if there is an isometry that sends x to y. Then our task becomes to find a perfect matching in the appropriate subgraph of this graph – namely, the bipartite subgraph where one partition contains points of the first ball and the other contains points of the other two balls. Banach and Tarski have shown that, with a suitably chosen set of isometries, the axiom of choice implies the existence of such a matching. In contrast, since isometries preserve the Lebesgue measure, the pieces in the decomposition cannot be Lebesgue measurable. Surprisingly, Dougherty and Foreman [45] proved that the pieces in the Banach-Tarski paradox can have the *Baire property*. The Baire property is a topological analogue of being Lebesgue measurable; a subset of $\mathbb{R}^3$ is said to have the Baire property if its difference from some open set is topologically negligible.

Recently, results similar to the Banach-Tarski paradox that lie on the border of combinatorics, logic, group theory, and ergodic theory led to an emergence of a new field often called *descriptive* or *measurable combinatorics*. The field focuses on the connection between the discrete and continuous and is largely concerned with the investigation of graph-theoretic concepts. The usual setup in descriptive combinatorics is that we have a graph with uncountably many connected components, each being a countable graph of bounded degree. For example, in case of the Banach-Tarski paradox, the vertices of the underlying graph are the points of the three balls, edges correspond to isometries, and the degree of each vertex is bounded by the number of chosen isometries. Some of the most beautiful results related to the field include [80, 87, 59, 45, 84, 51, 77, 86, 42, 38, 44, 20], see [76, 95] for recent surveys.

Importantly, in many of these results, including the Banach-Tarski paradox, graphs where each component is an infinite Δ-regular tree appear naturally. Oftentimes, questions considered in descriptive combinatorics lead to constructing a solution to a local problem in the underlying uncountable graph (in the case of Banach-Tarski, the local problem is perfect matching). The construction needs to be such that the solution of the problem has some additional regularity properties. For example in the case of Banach-Tarski, a solution is possible when the regularity condition is the Baire property, but not if it is Lebesgue measurability. In fact, together with Borel measurability these are the most prominent regularity conditions studied in descriptive combinatorics. The corresponding complexity classes of local problems that always admit a solution with

the respective regularity property are BOREL, MEASURE, BAIRE (See Figure 1). In this paper, we moreover consider the setting where each connected component of the underlying graph is a Δ-regular tree.

The connection between distributed computing and descriptive combinatorics arises from the fact that in descriptive combinatorics we care about constructions that do not use the axiom of choice. In the distributed language, the axiom of choice corresponds to leader election, that is, the constructions in descriptive combinatorics do not allow picking exactly one point in every component. To get some intuition about the power of the complexity class BOREL, we note that Borel constructions allow us to alternate countably many times the following two operations. First, any local algorithm with constant local complexity can be run. Second, we have an oracle that provides a maximal independent set (MIS) on any graph that can be constructed locally from the information computed so far [77]. Note that from the speedup result of [34] we get that every local problem with local complexity $O(\log^* n)$ can be solved by constant local constructions and *one* call to such an MIS oracle. This implies the inclusion $\mathsf{LOCAL}(O(\log^* n)) \subseteq \mathsf{BOREL}$ in Figure 1 proven in the insightful paper of Bernshteyn [20]. The relationship of the class MEASURE (and of the class fiid from the discussion of factors) to the class BOREL is analogous to the relationship of randomized distributed algorithms to deterministic distributed algorithms.

Local Problems on Regular Trees After introducing the three areas of interest in this work, we conclude the section by briefly discussing the kinds of problems we focus on, which are local problems on regular trees. More precisely, we study *locally checkable labeling (LCL)* problems, which are a class of problems, where the correctness of the solution can be checked locally. Examples include classical problems from combinatorics such as proper vertex coloring, proper edge coloring, perfect matching, and maximal independent set. One main goal of this paper is to understand possible complexity classes of LCLs without inputs on infinite Δ-regular trees and their finite analogues. We refer the reader to Section 3 for a precise definition of a finite Δ-regular tree.

The motivation for studying regular trees in this work stems from different sources: (a) infinite Δ-regular trees are studied in the area of ergodic theory [25, 24], random processes [2, 4, 83] and descriptive combinatorics [86, 39], (b) many lower bounds in distributed computing are proven in regular trees [8, 9, 27, 26, 31, 33, 58], and (c) connecting and comparing the techniques of the three areas in this simple setting reveals already deep connections, see Section 2.

2 Our Contributions

We believe that our main contribution is presenting all three perspectives as part of a common theory. Our technical contribution is split into three main parts.

2.1 Generalization of Marks' Technique

In Section 4 we extend the Borel determinacy technique of Marks [86], which was used to prove the nonexistence of Borel Δ-colorings and perfect matchings, to a broader class of problems, and adapt the extended technique to the distributed setting, thereby obtaining a simple method for proving distributed lower bounds. This method is the first lower bound technique for distributed computing using ideas coming from descriptive combinatorics (see [21] for a distributed computing upper bound motivated by descriptive combinatorics). Moreover, we show how to use the developed techniques to obtain both BOREL and LOCAL lower bounds for local problems from a natural class, called homomorphism problems. Our key technical ingredient for obtaining the mentioned techniques

and results is a novel technique based on the notion of an *ID graph*. We note that a very similar concept to the ID graph was independently discovered by [50].

Marks' technique In the following we give an introduction to Marks' technique by going through a variant of his proof [86, 85] that shows that Δ-coloring has deterministic local complexity $\Omega(\log n)$. The proof already works in the case where the considered regular tree comes with an input Δ-edge coloring. In this case, the output color of a given vertex u can be interpreted as that u "grabs" the incident edge of that color. The problem then reduces to the *edge grabbing* problem where every vertex is required to grab an incident edge such that no two vertices grab the same edge.

We first show the lower bound in the case that vertices do not have unique identifiers but instead are properly colored with $L > \Delta$ colors. Suppose there is an algorithm $\mathcal{A}$ solving the edge grabbing problem with local complexity $t(n) = o(\log n)$, and consider a tree rooted at vertex u of depth $t(n)$; such a tree has less than n vertices, for large enough n. Assume that u has input color $\sigma \in [L]$, and consider, for some fixed edge color α, the edge e that is incident to u and has color α. Two players, Alice and Bob, are playing the following game. In the i-th round, Alice colors the vertices at distance i from u in the subtree reachable via edge e with colors from $[L]$. Then, Bob colors all other vertices at distance i from u with colors from $[L]$ (see Figure 4). Consider the output of u when executing $\mathcal{A}$ on the obtained colored tree. Bob wins the game if u grabs the edge e, and Alice wins otherwise.

Note that either Alice or Bob has a winning strategy. Given the color σ of u, if, for each edge color α, Alice has a winning strategy in the game corresponding to the pair (σ, α), then we can create Δ copies of Alice and let them play their strategy on each subtree of u, telling them that the colors chosen by the other Alices are what Bob played. The result is a coloring of the input tree such that u, by definition, does not pick any edge, contradicting the fact that $\mathcal{A}$ provides a valid solution! So for every σ there is at least one α such that Bob has a winning strategy for the game corresponding to (σ, α). By the pigeonhole principle, there are two colors σ_1, σ_2, such that Bob has a winning strategy for both pairs (σ_1, α) and (σ_2, α). But now we can imagine a tree rooted in an edge between vertices u_1, u_2 that are colored with colors σ_1, σ_2. We can now take two copies of Bob, one playing at u_1 and the other playing at u_2 and let them battle it out, telling each copy that the other color from $\{\sigma_1, \sigma_2\}$ and whatever the other copy plays are the moves of Alice. The resulting coloring has the property that both u_1 and u_2, when executing $\mathcal{A}$ on the obtained colored tree, grab the edge between them, a contradiction that finishes the proof!

The ID graph The downside of the proof is that it does not work in the model with unique identifiers (where the players' moves consist in assigning identifiers instead of colors), since gluing copies of the same player could result in an identifier assignment where the identifiers are not unique. One possible remedy is to conduct the whole proof in the context of Borel graphs as was done by Marks. This proves an even stronger statement, namely that Δ-coloring is not in the class BOREL, but requires additional ad-hoc tricks and a pretty heavy set theoretic tool—Martin's celebrated Borel determinacy theorem [88] stating that even for infinite two-player games one of the players has to have a winning strategy if the payoff set is Borel. The ID graph enables us to adapt the proof (and its generalization that we develop in Section 4) to the distributed setting, where the fact that one of the players has a winning strategy is obvious. Moreover, we use an infinite version of the ID graph to generalize Marks' technique also in the Borel setting.

Here is how it works: The ID graph is a specific graph whose vertices are the possible unique input identifiers (for input graphs of size n), that is, numbers from $[n^{O(1)}]$. Its edges are colored with colors from $[\Delta]$ and its girth is $\Omega(\log n)$. When we define the game between Alice and Bob, we

require them to label vertices with identifiers in such a way that whenever a new vertex is labeled with identifier i, and its already labeled neighbor has identifier j, then ij is an edge in the ID graph. Moreover, the color of edge ij in the ID graph is required to be the same as the color of the edge between the vertices labeled i and j in the tree where the game is played. It is straightforward to check that these conditions enforce that even if we let several copies of the same player play, the resulting tree is labeled with unique identifiers. Hence, the same argument as above now finally proves that the deterministic local complexity of Δ-coloring is $\Omega(\log n)$.

We note that our ID graph technique is of independent interest and may have applications in many different contexts. To give an example from distributed computing, consider the proof of the deterministic $\Omega(\log n)$-round lower bound for Δ-coloring developed by the distributed community [27, 34], which is based on the celebrated round elimination technique. Even though the result is deterministic, the proof is quite technical due to the fact that it relies on examining randomized algorithms, for reasons similar to the reasons why Marks' proof does not apply directly to the setting with unique identifiers. Fortunately, it can be again streamlined with the use of the ID graph technique. Moreover, the ID graph technique has already led to a new lower bound for the Lovász local lemma [29] in the area of Local Computation Algorithms (which is part of the realm of sequential computation), thereby giving further evidence for the versatility of the technique.

Marks vs. Round Elimination It is quite insightful to compare Marks' technique (and our generalization of it) with the powerful round elimination technique [26], which has been responsible for all locality lower bounds of the last years [27, 9, 26, 15, 8, 31, 11, 33, 10]. While, on the surface, Marks' approach developed for the Borel world may seem quite different from the round elimination technique, there are actually striking similarities between the two methods. On a high level, in the round elimination technique, the following argument is used to prove lower bounds in the LOCAL model: If a T-round algorithm exists for a problem Π_0 of interest, then there exists a $(T-1)$-round algorithm for some problem Π_1 that can be obtained from Π_0 in a mechanical manner. By applying this step iteratively, we obtain a problem Π_t that can be solved in 0 rounds; by showing that there is no 0-algorithm for Π_t (which is easy to do if Π_t is known), a $(T+1)$-round lower bound for Π_0 is obtained.

The interesting part regarding the relation to Marks' technique is how the $(T-i-1)$-round algorithms $\mathcal{A}'$ are obtained from the $(T-i)$-round algorithms $\mathcal{A}$ in the round elimination framework: in order to execute $\mathcal{A}'$, each vertex v, being aware of its $(T-i-1)$-hop neighborhood, essentially asks whether, for all possible extensions of its view by one hop along a chosen incident edge, there exists some extension of its view by one hop along all other incident edges such that $\mathcal{A}$, executed on the obtained $(T-i)$-hop neighborhood, returns a certain output at v, and then bases its output on the obtained answer. It turns out that the vertex sets corresponding to these two extensions correspond precisely to two moves of the two players in the game(s) played in Marks' approach: more precisely, in round $T-i$ of a game corresponding to the considered vertex v and the chosen incident edge, the move of Alice consists in labeling the vertices corresponding to the first extension, and the move of Bob consists in labeling the vertices corresponding to the second extension.

However, despite the similarities, the two techniques (at least in their current forms) have their own strengths and weaknesses and are interestingly different in that there are local problems that we know how to obtain good lower bounds for with one technique but not the other, and vice versa. Finding provable connections between the two techniques is an exciting research direction that we expect to lead to a better understanding of the possibilities and limitations of both techniques.

In Section 4 we use our generalized and adapted version of Marks' technique to prove new lower bounds for so-called homomorphism problems. Homomorphism problems are a class of local

problems that generalizes coloring problems—each vertex is to be colored with some color and there are constraints on which colors are allowed to be adjacent. The constraints can be viewed as a graph—in the case of coloring this graph is a clique. In general, whenever the underlying graph[1] of the homomorphism problem is Δ-colorable, its deterministic local complexity is $\Omega(\log n)$, because solving the problem would imply that we can solve Δ-coloring too (in the same runtime). It seems plausible that homomorphism problems of this kind are the only hard, i.e., $\Omega(\log n)$, homomorphism problems. However, our generalization of Marks' technique asserts that this is not true.

Theorem 1. *There are homomorphism problems whose deterministic local complexity on trees of degree $\leq \Delta$ is $\Omega(\log n)$ such that the chromatic number of the underlying graph is $2\Delta - 2$.*

It is not known how to prove the same lower bounds using round elimination[2]; in fact, as far as we know, these problems are the only known examples of problems on Δ-regular trees for which a lower bound is known to hold but currently not achievable by round elimination. Proving the same lower bounds via round elimination is an exciting open problem.

2.2 Separation of Various Complexity Classes

Uniform Complexity Landscape We investigate the connection between randomized and uniform distributed local algorithms, where uniform algorithms are equivalent to the studied notion of finitary factors of iid. First, it is simple to observe that local problems with uniform complexity $t(\varepsilon)$ have randomized complexity $t(1/n^{O(1)})$ – by definition, every vertex knows its local output after that many rounds with probability $1 - 1/n^{O(1)}$. The result thus follows by a union bound over the n vertices of the input graph.

On the other hand, we observe that on Δ-regular trees the implication also goes in the opposite direction in the following sense. Every problem that has a randomized complexity of $t(n) = o(\log n)$ has a uniform complexity of $O(t(1/\varepsilon))$.

One could naively assume that this equivalence also holds for higher complexities, but this is not the case. Consider for example the 3-coloring problem. It is well-known in the distributed community that 3-coloring a tree can be solved deterministically in $O(\log n)$ rounds using the rake-and-compress decomposition [35, 92]. On the other hand, there is no uniform algorithm for 3-coloring a tree. If there were such a uniform algorithm, we could run it on any graph with large enough girth and color 99% of its vertices with three colors. This in turn would imply that the high-girth graph has an independent set of size at least $0.99 \cdot n/3$. This is a contradiction with the fact that there exist high-girth graphs with a much smaller independence number [23].

Interestingly, the characterization of Bernshteyn [19] implies that *any* uniform distributed algorithm can be "sped up" to a deterministic local $O(\log n)$ complexity, as we prove in Theorem 16.

We show that there are local problems that can be solved by a uniform algorithm but only with a complexity of $\Omega(\log 1/\varepsilon)$. Namely, the problem of constructing a 2-hop perfect matching on infinite Δ-regular trees for $\Delta \geq 3$ has a uniform local complexity between $\Omega(\log 1/\varepsilon)$ and $O(\operatorname{poly}\log 1/\varepsilon)$. Formally, this proves the following theorem.

Theorem 2. $\mathsf{ULOCAL}(O(\log\log 1/\varepsilon)) \subsetneq \mathsf{ULOCAL}(O(\operatorname{poly}\log 1/\varepsilon))$.

[1] Note that the maximum degree of the underlying graph is potentially very different from Δ, the maximum degree of the input tree.

[2] Indeed, the descriptions of the problems have comparably large numbers of labels and do not behave like so-called "fixed points" (i.e., nicely) under round elimination, which suggests that it is hard to find a round elimination proof with the currently known approaches.

The uniform algorithm for this problem is based on a so-called one-ended forest decomposition introduced in [39] in the descriptive combinatorics context. In a one-ended forest decomposition, each vertex selects exactly one of its neighbors as its parent by orienting the corresponding edge outwards. This defines a decomposition of the vertices into infinite trees. We refer to such a decomposition as a one-ended forest decomposition if the subtree rooted at each vertex only contains finitely many vertices. Having computed such a decomposition, 2-hop perfect matching can be solved inductively starting from the leaf vertices of each tree.

We leave the understanding of the uniform complexity landscape in the regime $\Omega(\log 1/\varepsilon)$ as an exciting open problem. In particular, does there exist a function $g(\varepsilon)$ such that each local problem that can be solved by a uniform algorithm has a uniform complexity of $O(g(\varepsilon))$?

Relationship of Distributed Classes with Descriptive Combinatorics Bernshteyn recently proved that $\mathsf{LOCAL}(O(\log^* n)) \subseteq \mathsf{BOREL}$ [20]. That is, each local problem with a deterministic LOCAL complexity of $O(\log^* n)$ also admits a Borel-measurable solution. A natural question to ask is whether the converse also holds. Indeed, it is known that $\mathsf{LOCAL}(O(\log^* n)) = \mathsf{BOREL}$ on paths with no additional input [61]. We show that on regular trees the situation is different. On one hand, a characterization of Bernshteyn [19] implies that $\mathsf{BOREL} \subseteq \mathsf{BAIRE} \subseteq \mathsf{LOCAL}(O(\log n))$. On the other hand, we show that this result cannot be strengthened by proving the following result.

Theorem 3. $\mathsf{BOREL} \not\subseteq \mathsf{RLOCAL}(o(\log n))$.

That is, there exists a local problem that admits a Borel-measurable solution but cannot be solved with a (randomized) LOCAL algorithm running in a sublogarithmic number of rounds.

Let us sketch a weaker separation, namely that $\mathsf{BOREL} \setminus \mathsf{LOCAL}(O(\log^* n)) \neq \emptyset$. Consider a version of Δ-coloring where a subset of vertices can be left uncolored. However, the subgraph induced by the uncolored vertices needs to be a collection of doubly-infinite paths (in finite trees, this means each path needs to end in a leaf vertex). The nonexistence of a fast distributed algorithm for this problem essentially follows from the celebrated $\Omega(\log n)$ deterministic lower bound for Δ-coloring of [27]. On the other hand, the problem allows a Borel solution. First, sequentially compute $\Delta - 2$ maximal independent sets, each time coloring all vertices in the MIS with the same color, followed by removing all the colored vertices from the graph. In that way, a total of $\Delta - 2$ colors are used. Moreover, each uncolored vertex has at most 2 uncolored neighbors. This implies that the set of uncolored vertices forms a disjoint union of finite paths, one ended infinite paths and doubly infinite paths. The first two classes can be colored inductively with two additional colors, starting at one endpoint of each path in a Borel way (namely it can be done by making use of the countably many MISes in larger and larger powers of the input graph). Hence, in the end only doubly infinite paths are left uncolored, as desired.

To show the stronger separation between the classes BOREL and $\mathsf{RLOCAL}(o(\log n))$ we use a variation of the 2-hop perfect matching problem. In this variation, some of the vertices can be left unmatched, but similar as in the variant of the Δ-coloring problem described above, the graph induced by all the unmatched vertices needs to satisfy some additional constraints.

We conclude the paragraph by noting that the separation between the classes BOREL and $\mathsf{LOCAL}(O(\log^* n))$ is not as simple as it may look in the following sense. This is because problems typically studied in the LOCAL model with a LOCAL complexity of $\omega(\log^* n)$ like Δ-coloring and perfect matching also do not admit a Borel-measurable solution due to the technique of Marks [86] that we discussed in Section 2.1.

2.3 LOCAL($O(\log n)$) = BAIRE

We already discussed that one of complexity classes studied in descriptive combinatorics is the class BAIRE. Recently, Bernshteyn proved [19] that all local problems that are in the complexity class BAIRE, MEASURE or fiid have to satisfy a simple combinatorial condition which we call being ℓ-full. On the other hand, all ℓ-full problems allow a BAIRE solution [19]. This implies a complete combinatorial characterization of the class BAIRE. We defer the formal definition of ℓ-fullness to Section 6 as it requires a formal definition of a local problem. Informally speaking, in the context of vertex labeling problems, a problem is ℓ-full if we can choose a subset S of the labels with the following property. Whenever we label two endpoints of a path of at least ℓ vertices with two labels from S, we can extend the labeling with labels from S to the whole path such that the overall labeling is valid. For example, proper 3-coloring is 3-full with $S = \{1, 2, 3\}$ because for any path of three vertices such that its both endpoints are colored arbitrarily, we can color the middle vertex so that the overall coloring is proper. On the other hand, proper 2-coloring is not ℓ-full for any ℓ.

We complement this result as follows. First, we prove that any ℓ-full problem has local complexity $O(\log n)$, thus proving that all complexity classes considered in the areas of factors of iids and descriptive combinatorics from Figure 1 are contained in LOCAL($O(\log n)$). In particular, this implies that the existence of *any* uniform algorithm implies a local distributed algorithm for the same problem of local complexity $O(\log n)$. We obtain this result via the well-known rake-and-compress decomposition [92].

On the other hand, we prove that any problem in the class LOCAL($O(\log n)$) satisfies the ℓ-full condition. The proof combines a machinery developed by Chang and Pettie [35] with additional nontrivial ideas. In this proof we construct recursively a sequence of sets of rooted, layered, and partially labeled trees, where the partial labeling is computed by simulating any given $O(\log n)$-round distributed algorithm, and then the set S meeting the ℓ-full condition is constructed by considering all possible extensions of the partial labeling to complete correct labeling of these trees.

This result implies the following equality:

Theorem 4. LOCAL($O(\log n)$) = BAIRE.

This equality is surprising in that the definitions of the two classes do not seem to have much in common at first glance! Moreover, the proof of the equality relies on nontrivial results in both distributed algorithms (the technique of Chang and Pettie [35]) and descriptive combinatorics (the fact that a hierarchical decomposition, so-called toast, can be constructed in BAIRE, [41], see Proposition 12).

The combinatorial characterization of the local complexity class LOCAL($O(\log n)$) on Δ-regular trees is interesting from the perspective of distributed computing alone. This result can be seen as a part of a large research program aiming at classification of possible local complexities on various graph classes [13, 27, 34, 35, 32, 36, 12, 8, 14]. That is, we wish not only to understand possible complexity classes (see the left part of Figure 1 for possible local complexity classes on regular trees), but also to find combinatorial characterizations of problems in those classes that allow us to efficiently decide for a given problem which class it belongs to. Unfortunately, even for grids with input labels, it is *undecidable* whether a given local problem can be solved in $O(1)$ rounds [93, 30], since local problems on grids can be used to simulate a Turing machine. This undecidability result does not apply to paths and trees, hence for these graph classes it is still hopeful that we can find simple and useful characterizations for different classes of distributed problems.

In particular, on paths it is decidable what classes a given local problem belongs to, for all classes coming from the three areas considered here, and this holds even if we allow inputs [36, 61]. The situation becomes much more complicated when we consider trees. Recently, classification

results on trees were obtained for so-called binary-labeling problems [8]. More recently, a complete classification was obtained in the case of *rooted regular trees* [12]. Although their algorithm takes exponential time in the worst case, the authors provided a practical implementation fast enough to classify many typical problems of interest.

Much less is known for general, *unoriented* trees, with an arbitrary number of labels. In general, deciding the optimal distributed complexity for a local problem on bounded-degree trees is EXPTIME-hard [32], such a hardness result does not rule out the possibility for having a simple and polynomial-time characterization for the case of *regular trees*, where there is no input and the constraints are placed only on degree-Δ vertices. Indeed, it was stated in [12] as an open question to find such a characterization. Our characterization of $\mathsf{LOCAL}(O(\log n)) = \mathsf{BAIRE}$ by ℓ-full problems makes progress in better understanding the distributed complexity classes on trees and towards answering this open question.

Roadmap In Section 3, we define formally all the three setups we consider in the paper. In Section 4 we discuss the lower bound technique of Marks and the new concept of an ID graph. Next, in Section 5 we prove some basic results about the uniform complexity classes and give examples of problems separating some classes from Figure 1. Finally, in Section 6 we prove that a problem admits a Baire measurable solution if and only if it admits a distributed algorithm of local complexity $O(\log n)$.

The individual sections can be read largely independently of each other. Moreover, most of our results that are specific to only one of the three areas can be understood without reading the parts of the paper that concern the other areas. We encourage the reader interested mainly in one of the areas to skip the respective parts.

3 Preliminaries

In this section, we explain the setup we work with, the main definitions and results. The class of graphs that we consider in this work are either infinite Δ-regular trees, or their finite analogue that we define formally in Section 3.1.

We sometimes explicitly assume $\Delta > 2$. The case $\Delta = 2$, that is, studying paths, behaves differently and seems much easier to understand [61]. Unless stated otherwise, we do not consider any additional structure on the graphs, but sometimes it is natural to work with trees with an input Δ-edge-coloring.

3.1 Local Problems on Δ-regular trees

The problems we study in this work are locally checkable labeling (LCL) problems, which, roughly speaking, are problems that can be described via local constraints that have to be satisfied in a suitable neighborhood of each vertex. In the context of distributed algorithms, these problems were introduced in the seminal work by Naor and Stockmeyer [93], and have been studied extensively since. In the modern formulation introduced in [26], instead of labeling vertices or edges, LCL problems are described by labeling half-edges, i.e., pairs of a vertex and an incident edge. This formulation is very general in that it not only captures vertex and edge labeling problems, but also others such as orientation problems, or combinations of all of these types. Before we can provide this general definition of an LCL, we need to introduce some definitions. We start by formalizing the notion of a half-edge.

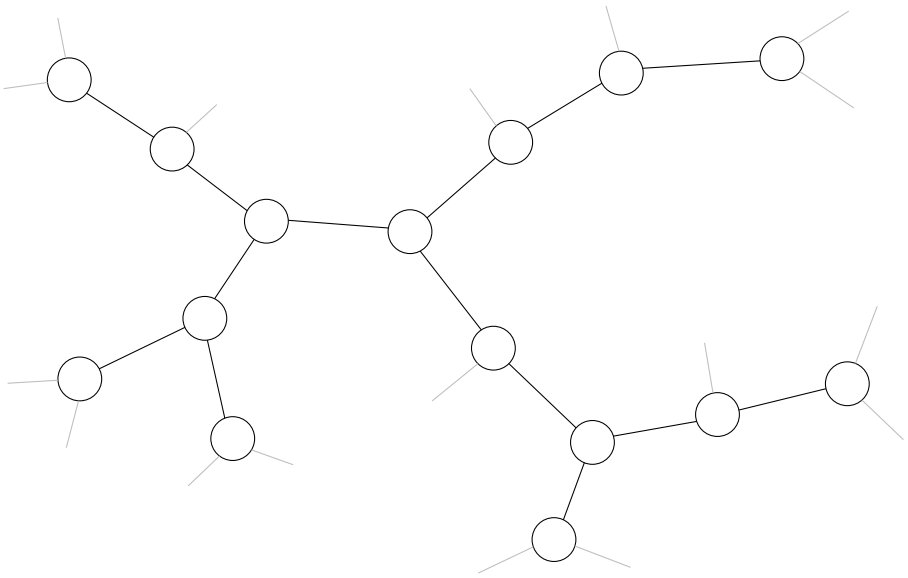

Figure 2: A 3-regular tree on 15 vertices. Every vertex has 3 half-edges but not all half-edges lead to a different vertex.

Definition 1 (Half-edge)**.** *A* half-edge *is a pair (v, e) where v is a vertex, and e an edge incident to v. We say that a half-edge (v, e) is* incident *to a vertex w if $w = v$, we say that a vertex w is* contained *in a half edge (v, e) if $w = v$, and we say that (v, e)* belongs *to an edge e' if $e' = e$. We denote the set of all half-edges of a graph G by $H(G)$. A* half-edge labeling *is a function $c\colon H(G) \to \Sigma$ that assigns to each half-edge an element from some label set Σ.*

In order to speak about finite Δ-regular trees, we need to consider slightly modified definition of a graph. We think of each vertex to be contained in Δ-many half-edges, however, not every half edge belongs to an actual edge of the graph. That is half-edges are pairs (v, e), but e is formally not a pair of vertices. Sometimes we refer to these half-edges as *virtual* half-edges. We include a formal definition to avoid confusions. See also Figure 2.

Definition 2 (Δ-regular trees)**.** *A tree T, finite or infinite is a* Δ-regular tree *if either it is infinite and $T = T_\Delta$, where T_Δ is the unique infinite Δ-regular tree, that is each vertex has exactly Δ-many neighbors, or it is finite of maximum degree Δ and each vertex $v \in T$ of degree $d \leq \Delta$ is contained in $(\Delta - d)$-many virtual half-edges.*

Formally, we can view T as a triplet $(V(T), E(T), H(T))$, where $(V(T), E(T))$ is a tree of maximum degree Δ and $H(T)$ consists of real half-edges, that is pairs (v, e), where $v \in V(T)$, $e \in E(T)$ and e is incident to v, together with some virtual edges, in the case when T is finite, such that each vertex $v \in V(T)$ is contained in exactly Δ-many half-edges (real or virtual).

As we are considering trees in this work, each LCL problem can be described in a specific form that provides two lists, one describing all label combinations that are allowed on the half-edges incident to a vertex, and the other describing all label combinations that are allowed on the two half-edges belonging to an edge.[3] We arrive at the following definition for LCLs on Δ-regular trees.[4]

[3]Every problem that can be described in the form given by Naor and Stockmeyer [93] can be equivalently described as an LCL problem in this list form, by simply requiring each output label on some half-edge h to encode all output labels in a suitably large (constant) neighborhood of h in the form given in [93].

[4]Note that the defined LCL problems do not allow the correctness of the output to depend on input labels.

Definition 3 (LCLs on Δ-regular trees)**.** *A* locally checkable labeling problem*, or* LCL *for short, is a triple* $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$*, where* Σ *is a finite set of labels,* $\mathcal{V}$ *is a subset of unordered cardinality-*Δ *multisets*[5] *of labels from* Σ *, and* $\mathcal{E}$ *is a subset of unordered cardinality-*2 *multisets of labels from* Σ*.*

We call $\mathcal{V}$ *and* $\mathcal{E}$ *the* vertex constraint *and* edge constraint *of* Π*, respectively. Moreover, we call each multiset contained in* $\mathcal{V}$ *a* vertex configuration *of* Π*, and each multiset contained in* $\mathcal{E}$ *an* edge configuration *of* Π*.*

Let T *be a* Δ*-regular tree and* $c : H(T) \to \Sigma$ *a half-edge labeling of* T *with labels from* Σ*. We say that* c *is a* Π-coloring*, or, equivalently, a correct solution for* Π*, if, for each vertex* v *of* T*, the multiset of labels assigned to the half-edges incident to* v *is contained in* $\mathcal{V}$*, and, for each edge* e *of* T*, the cardinality-*2 *multiset of labels assigned to the half-edges belonging to* e *is an element of* $\mathcal{E}$*.*

An equivalent way to define our setting would be to consider Δ-regular trees as commonly defined, that is, there are vertices of degree Δ and vertices of degree 1, i.e., leaves. In the corresponding definition of LCL one would consider leaves as unconstrained w.r.t. the vertex constraint, i.e., in the above definition of a correct solution the condition "for each vertex v" is replaced by "for each non-leaf vertex v". Equivalently, we could allow arbitrary trees of maximum degree Δ as input graphs, but, for vertices of degree $< \Delta$, we require the multiset of labels assigned to the half-edges to be extendable to some cardinality-Δ multiset in $\mathcal{V}$. When it helps the exposition of our ideas and is clear from the context, we may make use of these different but equivalent perspectives.

We illustrate the difference between our setting and "standard setting" without virtual half-edges on the perfect matching problem. A standard definition of the perfect matching problem is that some edges are picked in such a way that each vertex is covered by exactly one edge. It is easy to see that there is no local algorithm to solve this problem on the class of finite trees (without virtual half-edges), this is a simple parity argument. However, in our setting, every vertex needs to pick exactly one half-edge (real or virtual) in such a way that both endpoints of each edge are either picked or not picked. We remark that in our setting it is not difficult to see that (if $\Delta > 2$), then this problem can be solved by a local deterministic algorithm of local complexity $O(\log(n))$.

3.2 The LOCAL model

In this section, we define local algorithms and local complexity. We discuss the general relation between the classical local complexity and the uniform local complexity. Recall that when we talk about distributed algorithm on Δ-regular trees, the algorithm has access to n, the size of the tree. The measure of complexity is the classical local complexity. On the other hand, when we talk about uniform distributed algorithms on Δ-regular trees, we talk about an infinite Δ-regular tree and the measure of complexity is the uniform local complexity. We start with the classical notions. Recall that $B(v, t)$ is the t-hop neighborhood of a vertex v in an underlying graph G.

Definition 4 (Local algorithm)**.** *A* distributed local algorithm $\mathcal{A}$ *of local complexity* $t(n)$ *is a function defined on all possible* $t(n)$*-hop neighborhoods of a vertex. Applying an algorithm* $\mathcal{A}$ *on an input graph* G *means that the function is applied to a* $t(n)$*-hop neighborhood* $B(u, t(n))$ *of each vertex* u *of* G*. The output of the function is a labeling of the half-edges around the given vertex. The algorithm also takes as input the size of the input graph* n*.*

Definition 5 (Local complexity)**.** *We say that an LCL problem* Π *has a deterministic local complexity* $t(n)$ *if there is a local algorithm* $\mathcal{A}$ *of local complexity* $t(n)$ *such that when run on the input graph* G*, with each of its vertices having a unique identifier from* $[n^{O(1)}]$*,* $\mathcal{A}$ *always returns a valid solution to* Π*. We also say* $\Pi \in \mathsf{DLOCAL}(O(t(n)))$*.*

[5]Recall that a multiset is a modification of the concept of sets, where repetition is allowed.

The randomized complexity is defined analogously, but instead of unique identifiers, each vertex of G has an infinite random string. The solution of $\mathcal{A}$ needs to be a valid solution with probability $1 - 1/n$. We also say $\Pi \in \mathsf{RLOCAL}(O(t(n))$.

Whenever we talk about *local* complexity on Δ-regular trees, we always tacitly think about the class of finite Δ-regular trees. We use the notation $\mathsf{LOCAL}(O(t(n)))$ whenever the deterministic and the randomized complexity of the problems are the same up to constant factor (cf. Theorem 5).

Classification of Local Problems on Δ-regular Trees There is a lot of work aiming to classify all possible local complexities of LCL problems on bounded degree trees. This classification was recently finished. Even though the results in the literature are stated for the classical notion of finite trees, we note that it is easy to check that the same picture emerges if we restrict ourselves to Δ-regular trees according to our definition.

Theorem 5 (Classification of local complexities of LCLs on Δ-regular trees [93, 34, 35, 32, 13, 15, 28])**.** *Let Π be an LCL problem. Then the deterministic/randomized local complexity of Π on Δ-regular trees is one of the following:*

1. $O(1)$,
2. $\Theta(\log^* n)$,
3. $\Theta(\log n)$ *deterministic and* $\Theta(\log \log n)$ *randomized,*
4. $\Theta(\log n)$,
5. $\Theta(n^{1/k})$ *for* $k \in \mathbb{N}$.

We do not know whether the complexity classes $\Theta(n^{1/k})$ are non-empty on Δ-*regular* trees. The $2\frac{1}{2}$ coloring problems $\mathcal{P}_k$ are known to have complexity $\Theta(n^{1/k})$ on *bounded-degree* trees [35]. However, as the correctness criterion of $\mathcal{P}_k$ depends on the degree of vertices, it does not fit into the class of LCL problems that we consider in Definition 3. Nevertheless, as can be seen in Figure 1, all classes of factors of iid and from descriptive combinatorics are already contained in the class $\mathsf{LOCAL}(O(\log n))$.

Uniform Algorithms As we mentioned before, when talking about local complexities, we always have in mind that the underlying graph is finite. In particular, the corresponding algorithm knows the size of the graph. On infinite Δ-regular trees, or infinite graphs in general, we use the following notion [72, 78].

Definition 6. *An* uniform local algorithm $\mathcal{A}$ *is a function that is defined on all possible (finite) neighborhoods of a vertex. For some neighborhoods it outputs a special symbol $\emptyset$ instead of a labeling of the half-edges around the central vertex. Applying $\mathcal{A}$ on a graph G means that for each vertex u of G the function is applied to $B(u,t)$, where t is the minimal number such that $\mathcal{A}(u,t) \neq \emptyset$. We call t the coding radius of $\mathcal{A}$, and denote it, as a function on vertices, as $R_{\mathcal{A}}$.*

We define the corresponding notion of *uniform local complexity* for infinite Δ-regular trees where each vertex is assigned an infinite random string.

Definition 7 (Uniform local complexity [72])**.** *We say that the* uniform local (randomized) complexity *of an LCL problem Π is $t(\varepsilon)$ if there is a uniform local algorithm $\mathcal{A}$ such that the following hold on the infinite Δ-regular tree. Recall that $R_{\mathcal{A}}$ is the random variable measuring the coding*

radius of a vertex u, that is, the distance $\mathcal{A}$ needs to look at to decide the answer for u. Then, for any $0 < \varepsilon < 1$:

$$P(R_{\mathcal{A}} \geq t(\varepsilon)) \leq \varepsilon.$$

We also say $\Pi \in \mathsf{OLOCAL}(O(t(\varepsilon))$.

We finish by stating the following lemma that bounds the uniform complexity of concatenation of two uniform algorithm (we need to be a little bit more careful and cannot just add the complexites up).

Lemma 1 (Sequential Composition)**.** *Let A_1 and A_2 be two distributed uniform algorithms with a uniform local complexity of $t_1(\varepsilon)$ and $t_2(\varepsilon)$, respectively. Let A be the sequential composition of A_1 and A_2. That is, A needs to know the output that A_2 computes when the local input at every vertex is equal to the output of the algorithm A_1 at that vertex. Then, $t_A(\varepsilon) \leq t_{A_1}\left(\frac{\varepsilon/2}{\Delta^{t_{A_2}(\varepsilon/2)+1}}\right) + t_{A_2}(\varepsilon/2)$.*

Proof. Consider some arbitrary vertex u. Let E_1 denote the event that the coding radius of $\mathcal{A}_1$ is at most $t_{A_1}\left(\frac{\varepsilon/2}{\Delta^{t_{A_2}(\varepsilon/2)+1}}\right)$ for all vertices in the $t_{A_2}(\varepsilon/2)$-hop neighborhood around u. As the $t_{A_2}(\varepsilon/2)$-hop neighborhood around u contains at most $\Delta^{t_{A_2}(\varepsilon/2)+1}$ vertices, a union bound implies that $P(E_1) \geq 1 - \varepsilon/2$. Moreover, let E_2 denote the event that the coding radius of algorithm $\mathcal{A}_2$ at vertex u is at most $t_{A_2}(\varepsilon/2)$. By definition, $P(E_2) \geq 1 - \varepsilon/2$. Moreover, if both events E_1 and E_2 occur, which happens with probability at least $1 - \varepsilon$, then the coding radius of algorithm $\mathcal{A}$ is at most $t_{A_1}\left(\frac{\varepsilon/2}{\Delta^{t_{A_2}(\varepsilon/2)+1}}\right) + t_{A_2}(\varepsilon/2)$, thus finishing the proof. □

3.3 Descriptive combinatorics

Before we define formally the descriptive combinatorics complexity classes, we give a high-level overview on their connection to distributing computing for the readers more familiar with the latter.

The complexity class that partially captures deterministic local complexity classes is called BOREL (see also Remark 1). First note that by a result of Kechris, Solecki and Todorčević [77] the *maximal independent set problem* (with any parameter $r \in \mathbb{N}$) is in this class for any bounded degree graph.[6] In particular, this yields that BOREL contains the class $\mathsf{LOCAL}(O(\log^* n))$ by the characterization of [34], see [20]. Moreover, as mentioned before, BOREL is closed under countably many iterations of the operations of finding maximal independent set (for some parameter that might grow) and of applying a constant local rule that takes into account what has been constructed previously.[7]

To get a better grasp of what this means, consider for example the proper vertex 2-coloring problem on half-lines. It is clear that no local algorithm can solve this problem. However, as it is possible to determine the starting vertex after countably many iterations of the maximal independent set operation, we conclude that this problem is in the class BOREL. The idea that BOREL can compute some unbounded, global, information will be implicitly used in all the constructions in Section 5 that separate BOREL from local classes.

The intuition behind the class MEASURE is that it relates in the same way to the class BOREL, as randomized local algorithms relate to deterministic ones. In particular, the operations that are

[6] That is, it is possible to find a Borel maximal independent set, i.e., a maximal independent set which is, moreover, a Borel subset of the vertex set.

[7] It is in fact an open problem, whether this captures fully the class BOREL. However, note that an affirmative answer to this question would yield that problems can be solved in an "effective" manner in the Borel context, which is known not to be the case in unbounded degree graphs [106].

allowed in the class MEASURE are the same as in the class BOREL but the solution of a given LCL can be incorrect on a measure zero set.

The class BAIRE can be considered as a topological equivalent of the measure theoretic class MEASURE, that is, a solution can be incorrect on a topologically negligible set. The main difference between the classes MEASURE and BAIRE is that in the later there is a hierarchical decomposition that is called *toast*. (Note that this phenomenon is present in the case of MEASURE exactly on so-called amenable graphs. It is also tightly connected with the notion of hyperfiniteness [41, 55].) The independence of colorings on a tree together with this structure allows for a combinatorial characterization of the class BAIRE, which was proven by Bernshteyn [19], see also Section 6.

Next we formulate the precise definitions. We refer the reader to [95, 76, 20, 75], or to [61, Introduction, Section 4.1] and [62, Section 7.1, 7.2] for intuitive examples and standard facts of descriptive set theory. In particular, we do not define here the notion standard Borel/probability space, a Polish topology, a Borel probability measure, Baire property etc.

Let $\mathcal{G}$ be a Borel graph of bounded maximum degree on a standard Borel space X. In this paper we consider exclusively acyclic Δ-regular Borel graphs and we refer to them as *Δ-regular Borel forests*. It is easy to see that the set of half-edges (see Definition 1) is naturally a standard Borel space, we denote this set by $H(\mathcal{G})$. Thus, it makes sense to consider Borel labelings of $H(\mathcal{G})$. Moreover, if $\mathcal{G}$ is a Δ-regular Borel forest and $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ is an LCL, we can also decide whether a coloring $f : \mathcal{H}(\mathcal{G}) \to \Sigma$ is a solution to Π as in Definition 3. Similarly, we say that the coloring f solves Π, e.g., on a μ-conull set for some Borel probability measure μ on X if there is a Borel set $C \subseteq X$ such that $\mu(C) = 1$, the vertex constraints are satisfied around every $x \in C$ and the edge constraints are satisfied for every $x, y \in C$ that form an edge in $\mathcal{G}$.

Definition 8 (Descriptive classes)**.** *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL. We say that Π is in the class* BOREL *if for every acyclic Δ-regular Borel graph $\mathcal{G}$ on a standard Borel space X, there is a Borel function $f : H(\mathcal{G}) \to \Sigma$ that is a Π-coloring of $\mathcal{G}$.*

We say that Π is in the class BAIRE *if for every acyclic Δ-regular Borel graph $\mathcal{G}$ on a standard Borel space X and every compatible Polish topology τ on X, there is a Borel function $f : H(\mathcal{G}) \to \Sigma$ that is a Π-coloring of $\mathcal{G}$ on a τ-comeager set.*

We say that Π is in the class MEASURE *if for every acyclic Δ-regular Borel graph $\mathcal{G}$ on a standard Borel space X and every Borel probability measure μ on X, there is a Borel function $f : H(\mathcal{G}) \to \Sigma$ that is a Π-coloring of $\mathcal{G}$ on a μ-conull set.*

The following claim follows directly from the definition.

Claim 1. *We have* BOREL $\subseteq$ MEASURE, BAIRE.

Recently Bernshteyn [20, 22] proved several results that connect distributed computing with descriptive combinatorics. Using the language of complexity classes we can formulate some of the result as inclusions in Figure 1.

Theorem 6 ([20])**.** *We have*

- $\mathsf{LOCAL}(O(\log^*(n))) \subseteq \mathsf{BOREL}$,
- $\mathsf{RLOCAL}(o(\log(n))) \subseteq \mathsf{MEASURE}, \mathsf{BAIRE}$.

In fact, these inclusions hold for any reasonable class of bounded degree graphs.

Remark 1. *The "truly" local class in descriptive combinatorics is the class* CONTINUOUS*. Even though we do not define this class here*[8]*, and we refer the reader to [22, 56, 62] for the definition and discussions in various cases, we mention that the inclusion*

$$\mathsf{LOCAL}(O(\log^* n)) \subseteq \mathsf{CONTINUOUS} \qquad (*)$$

holds in most reasonable classes of bounded degree graphs, see [20]. This also applies to our situation. Recently, it was shown by Bernshteyn [22], and independently by Seward [99], that $(*)$ *can be reversed for Cayley graphs of finitely generated groups. This includes, e.g., natural* Δ*-regular trees with proper edge* Δ*-coloring, as this is a Cayely graph of free product of* Δ*-many* $\mathbb{Z}_2$ *induced by the standard generating set. It is, however, not clear whether it* $(*)$ *can be reversed in our situation, i.e.,* Δ*-regular trees without any additional labels.*

3.4 Random processes

We start with an intutitve description of fiid processes. Let T_Δ be the infinite Δ-regular tree. Informally, *factors of iid processes (fiid)* on T_Δ are infinite analogues of local randomized algorithms in the following way. Let Π be an LCL and $u \in T_\Delta$. In order to solve Π, we are allowed to explore the whole graph, and the random strings that are assigned to vertices, and then output a labeling of half-edges around u. If such an assignment is a measurable function and produces a Π-coloring almost surely, then we say that Π is in the class fiid. Moreover, if every vertex needs to explore only finite neighborhood to output a solution, then we say that Π is in the class ffiid. Such processes are called *finitary* fiid (*ffiid*). There is also an intimate connection between ffiid and uniform algorithms. This is explained in [62, Section 2.2]. Informally, an ffiid process that almost surely solves Π is, in the language of distributed computing, a uniform local algorithm that solves Π. This allows us to talk about uniform local complexity of an ffiid. In the rest of the paper we interchange both notions freely with slight preference for the distributed computing language.

Now we define formally these classes, using the language of probability. We denote by $\mathrm{Aut}(T_\Delta)$ the automorphism group of T_Δ. An *iid process* on T_Δ is a collection of iid random variables $Y = \{Y_v\}_{v \in V(T_\Delta)}$ indexed by vertices, or edges, half-edges etc, of T_Δ such that their joint distribution is invariant under $\mathrm{Aut}(T_\Delta)$. We say that X is a *factor of iid process (fiid)* if $X = F(Y)$, where F is a measurable $\mathrm{Aut}(T_\Delta)$-invariant map and Y is an iid process on T_Δ.[9] Moreover, we say that X is a *finitary factor if iid process (ffiid)* if F depends with probability 1 on a finite (but random) radius around each vertex. We denote as R_F the random variable that assigns minimal such radius to a given vertex, and call it the *coding radius* of F. We denote the space of all Π-colorings of T_Δ as X_Π. This is a subspace of $\Sigma^{H(T_\Delta)}$ that is invariant under $\mathrm{Aut}(T_\Delta)$.

Definition 9. *We say that an LCL* Π *is in the class* fiid *(*ffiid*) if there is an fiid (ffiid) process* X *that almost surely produces elements of* X_Π*.*

Equivalently, we can define $\mathsf{ffiid} = \bigcup_f \mathsf{OLOCAL}(f(\varepsilon))$ *where* f *ranges over all functions. That is,* ffiid *is the class of problems solvable by any uniform distributed algorithm.*

It is obvious that $\mathsf{ffiid} \subseteq \mathsf{fiid}$. The natural connection between descriptive combinatorics and random processes is formulated by the inclusion $\mathsf{MEASURE} \subseteq \mathsf{fiid}$. While this inclusion is trivially satisfied, e.g., in the case of Δ-regular trees with proper edge Δ-coloring, in our situation we need a routine argument that we include for the sake of completeness in Appendix A.

[8] To define the class CONTINUOUS, rather than asking for a continuous solution on all possible Δ-regular Borel graphs, one has to restrict to a smaller family of graphs, otherwise the class trivializes. To define precisely this family is somewhat inconvenient, and not necessary for our arguments.

[9] We always assume $Y = (2^{\mathbb{N}})^{T_\Delta}$ endowed with the product Lebesgue measure.

Lemma 2. *Let* Π *be an LCL such that* $\Pi \in \mathsf{MEASURE}$. *Then* $\Pi \in \mathsf{fiid}$.

3.5 Specific Local Problems

Here we list some well-known local problems that were studied in all three considered areas. We describe the results known about these problems with respect to classes from Figure 1.

Edge Grabbing We start by recalling the well-known problem of edge grabbing (a close variant of the problem is known as sinkless orientation[27]). In this problem, every vertex should mark one of its half-edges (that is, grab an edge) in such a way that no edge can be marked by two vertices. It is known that $\Pi_{\text{edgegrab}} \notin \mathsf{LOCAL}(O(\log^* n))$ [27] but $\Pi_{\text{edgegrab}} \in \mathsf{RLOCAL}(O(\log \log n))$.

Similarly, $\Pi_{\text{edgegrab}} \notin \mathsf{BOREL}$ by [86], but $\Pi_{\text{edgegrab}} \in \mathsf{MEASURE}$ by [39]: to see the former, just note that if a Δ-regular tree admits proper Δ-colorings of both edges and vertices, every vertex can grab an edge of the same color as the color of the vertex. Thus, $\Pi_{\text{edgegrab}} \in \mathsf{BOREL}$ would yield that Δ-regular Borel forests with Borel proper edge-colorings admit a Borel proper Δ-coloring, contradicting [86].

This completes the complexity characterization of Π_{edgegrab} as well as the proper vertex Δ-coloring with respect to classes in Figure 1.

Perfect Matching Another notorious LCL problem, whose position in Figure 1 is, however, not completely understood, is the perfect matching problem Π_{pm}. Recall that the perfect matching problem Π_{pm} asks for a matching that covers all vertices of the input tree.[10] It is known that $\Pi_{\text{pm}} \in \mathsf{fiid}$ [83], and it is easy to see that $\Pi_{\text{pm}} \notin \mathsf{RLOCAL}(O(\log \log(n)))$ (we will prove a stronger result in Proposition 9). Marks proved [86] that it is not in BOREL, even when the underlying tree admits a Borel bipartition. It is not clear if Π_{pm} is in ffiid, nor whether it is in $\mathsf{MEASURE}$.

Graph Homomorphism We end our discussion with LCLs that correspond to graph homomorphisms (see also the discussion in Section 4). These are also known as edge constraint LCLs. Let G be a finite graph. Then we define Π_G to be the LCL that asks for a homomorphism from the input tree to G, that is, vertices are labeled with vertices of G in such a way that edge relations are preserved. There are not many positive results except for the class BAIRE. It follows from the result of Bernshteyn [19] (see Section 6) that $\Pi_G \in \mathsf{BAIRE}$ if and only if G is not bipartite. The main negative results can be summarized as follows. An easy consequence of the result of Marks [86] is that if $\chi(G) \leq \Delta$, then $\Pi_G \notin \mathsf{BOREL}$. In this paper, we describe examples of graphs of chromatic number up to $2\Delta - 2$ such that the corresponding homomorphism problem is not in BOREL, see Section 4. The theory of *entropy inequalities* see [2] implies that if G is a cycle on more than 9 vertices, then $\Pi_G \notin \mathsf{fiid}$.

4 Generalization of Marks' technique

In this section, we first develop a new way of proving lower bounds in the LOCAL model based on a generalization of a technique of Marks [86]. Then, we use ideas arising in the finitary setting—connected to the adaptation of Marks' technique—to obtain new results back in the Borel context. For an introduction to Marks' technique and a high-level discussion about the challenges in adapting the technique to the standard distributed setting as well as our solution via the new notion of an ID graph, we refer the reader to Section 2.1.

[10] In our formalism this means that around each vertex exactly one half-edge is picked.

The setting we consider in this section is Δ-regular trees that come with a proper Δ-edge coloring with colors from $[\Delta]$. All lower bounds discussed already hold under these restrictions (and therefore also in any more general setting).

Recall that an LCL $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ is given by specifying a list of allowed vertex configurations and a list of allowed edge configurations (see Definition 3). To make our lower bounds as widely applicable as possible, we replace the latter list by a separate list for each of the Δ edge colors; in other words, we consider LCLs where the correctness of the output is allowed to depend on the input that is provided by the edge colors. Hence, formally, in this section, an LCL is more generally defined: it is a tuple $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ where Σ and $\mathcal{V}$ are as before, while $\mathcal{E} = (\mathcal{E}_\alpha)_{\alpha\in[\Delta]}$ is now a Δ-tuple of sets $\mathcal{E}_\alpha$ consisting of cardinality-2 multisets. Similarly as before, a half-edge labeling (see Definition 8 for the Borel context) with labels from Σ is a correct solution for Π if, for each vertex v, the multiset of labels assigned to the half-edges incident to v is contained in $\mathcal{V}$, and, for each edge e, the multiset of labels assigned to the half-edges belonging to e is contained in $\mathcal{E}_\alpha$, where α is the color of the edge e.

The idea of our approach is to identify a condition that an LCL Π necessarily has to satisfy if it is solvable in $O(\log^* n)$ rounds in the LOCAL model. Showing that Π does not satisfy this condition then immediately implies an $\Omega(\log n)$ deterministic and $\Omega(\log\log n)$ randomized lower bound, by Theorem 5.

In order to define our condition we need to introduce the following notion.

Definition 10. *A* configuration graph *is a Δ-tuple $\mathbb{P} = (\mathbb{P}_\alpha)_{\alpha\in\Delta}$ of graphs, where the vertex set of each of the graphs $\mathbb{P}_\alpha$ is the set of subsets of Σ, and there is an edge in $\mathbb{P}_\alpha$ connecting two vertices S, T if and only if there are $\mathtt{a} \in S$ and $\mathtt{b} \in T$ such that $\{\mathtt{a}, \mathtt{b}\} \in \mathcal{E}_\alpha$.*

Note that loops are allowed. Naturally, we will consider any two vertices of different $\mathbb{P}_\alpha$ to be distinct, even if they correspond to the same subset of Σ.

Now we are set to define the aforementioned condition. The intuition behind the playability condition is the following: assume that there exists a local algorithm $\mathcal{A}$ that solves Π using the t neighbourhood of a given vertex. We are going to define a family of two player games. The game will depend on some $S \subseteq \Sigma$. Alice and Bob will assign labels (or IDs) to the vertices in the t-neighbourhood (in some way specified later on, depending on $\alpha \in [\Delta]$). When the assignment is complete, we evaluate $\mathcal{A}$ on the root of the obtained labelled graph, this way obtaining an element s of Σ, and decide who is the winner based on $s \in S$ or not. Naturally, it depends on S and α, which player has a winning strategy. This gives rise to a two coloring of vertices of $\mathbb{P}_\alpha$ by colors Alice and Bob. The failure of the playability condition will guarantee that using a strategy stealing argument one can derive a contradiction.

Definition 11 (Playability condition)**.** *We say that an LCL $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ with a configuration graph $\mathbb{P} = (\mathbb{P}_\alpha)_{\alpha\in\Delta}$ is* playable *if for every $\alpha \in [\Delta]$ there is a coloring Λ_α of the vertices of $\mathbb{P}_\alpha$ with two colors $\{\text{Alice}, \text{Bob}\}$ such that the following conditions are satisfied:*

(A) For any tuple $(S_\alpha)_{\alpha\in[\Delta]} \in V(\mathbb{P}_1) \times \cdots \times V(\mathbb{P}_\Delta)$ satisfying $\Lambda_\alpha(S_\alpha) = \text{Alice}$ for each $\alpha \in [\Delta]$, there exists an $\mathtt{a}_\alpha \notin S_\alpha$ such that $\{\mathtt{a}_\alpha\}_{\alpha\in\Delta} \in \mathcal{V}$, and

(B) for any $\alpha \in [\Delta]$, and any tuple $(S, T) \in V(\mathbb{P}_\alpha) \times V(\mathbb{P}_\alpha)$ satisfying $\Lambda_\alpha(S) = \Lambda_\alpha(T) = \text{Bob}$, we have that (S, T) is an edge of $\mathbb{P}_\alpha$.

Our aim in this section is to show the following general results.

Theorem 7. *Let Π be an LCL that is not playable. Then Π is not in the class $\mathsf{LOCAL}(O(\log^* n))$.*

Using ideas from the proof of this result, we can formulate an analogous theorem in the Borel context. Let us mention that while Theorem 7 is a consequence of Theorem 8 by [20], we prefer to state the theorems separately. This is because the proof of the Borel version of the theorem uses heavily complex results such as the Borel determinacy theorem, theory of local-global limits and some set-theoretical considerations about measurable sets. This is in a stark contrast with the proof of Theorem 7 that uses 'merely' the existence of large girth graphs and determinacy of finite games. However, the ideas surrounding these concepts are the same in both cases.

Theorem 8. *Let Π be an LCL that is not playable. Then Π is not in the class* BOREL.

The main application of these abstract results is to find lower bounds for *(graph) homomorphism LCLs* aka *edge constraint LCLs*. We already defined this class in Section 3.1 but we recall the definition in the formalism of this section to avoid any confusion. Let G be a finite graph. Then $\Pi_G = (\Sigma, \mathcal{V}, \mathcal{E})$ is defined by letting

1. $\mathcal{V} = \{(\mathtt{a}, \dots, \mathtt{a}) : \mathtt{a} \in \Sigma\}$,
2. $\Sigma = V(G)$,
3. $\forall \alpha \in [\Delta]$ $(\mathcal{E}_\alpha = E(G))$.

An LCL for which 1 holds is called a *vertex-LCL*. There is a one-to-one correspondence between vertex-LCLs for which $\forall \alpha, \beta$ $(\mathcal{E}_\alpha = \mathcal{E}_\beta)$ and LCLs of the form Π_G. (Indeed, to vertex-LCLs with this property one can associate a graph G whose vertices are the labels in Σ and where two vertices ℓ, ℓ' are connected by an edge if and only if $\{\ell, \ell'\} \in \mathcal{E}$.) Note that if Π is a vertex-LCL, then condition (A) in Definition 11 is equivalent to the statement that no tuple $\{S_\alpha\}_{\alpha \in \Delta}$ that satisfies the assumption of (A) can cover Σ, i.e., that $\Sigma \nsubseteq S_1 \cup \dots \cup S_\Delta$ for such a tuple. Moreover, if Π_G is a homomorphism LCL, then $\mathbb{P}_\alpha = \mathbb{P}_\beta$ for every $\alpha, \beta \in \Delta$.

The simplest example of a graph homomorphism LCL is given by setting G to be the clique on k vertices; the associated LCL Π_G is simply the problem of finding a proper k-coloring of the vertices of the input graph. Now we can easily derive Marks' original result from Theorem 8.

Corollary 1 (Marks [86])**.** *Let G be a finite graph that has chromatic number at most Δ. Then Π_G is not playable. In particular, there is a Δ-regular Borel forest that has Borel chromatic number $\Delta + 1$.*

Proof. Let $A_1, \dots, A_\Delta$ be some independent sets that cover G, and $\Lambda_1, \dots, \Lambda_\Delta$ arbitrary colorings of the vertices of $\mathbb{P}_1, \dots, \mathbb{P}_\Delta$, respectively, with colors from $\{\text{Alice}, \text{Bob}\}$. It follows that $\Lambda_\alpha(A_\alpha) = \text{Alice}$ for every $\alpha \in \Delta$, since otherwise condition (B) in Definition 11 is violated with $S = T = A_\alpha$. But then condition (A) does not hold. □

As our main application we describe graphs with chromatic number larger than Δ such that Π_G is not playable. This rules out the hope that the complexity of Π_G is connected with chromatic number being larger than Δ. In Section 4.1 we show the following. Note that the case $k = \Delta$ is Corollary 1.

Theorem 9. *Let $\Delta > 2$ and $\Delta < k \leq 2\Delta - 2$. There exists a graph G_k with $\chi(G_k) = k$, such that Π_{G_k} is not playable and $\Pi_{G_k} \in$ RLOCAL$(O(\log \log n))$. In particular, $\Pi_{G_k} \notin$ BOREL and $\Pi_{G_k} \notin$ LOCAL$(O(\log^*(n)))$.*

Interestingly, recent results connected to counterexamples to Hedetniemi's conjecture yield the same statement asymptotically, as $\Delta \to \infty$ (see Remark 4).

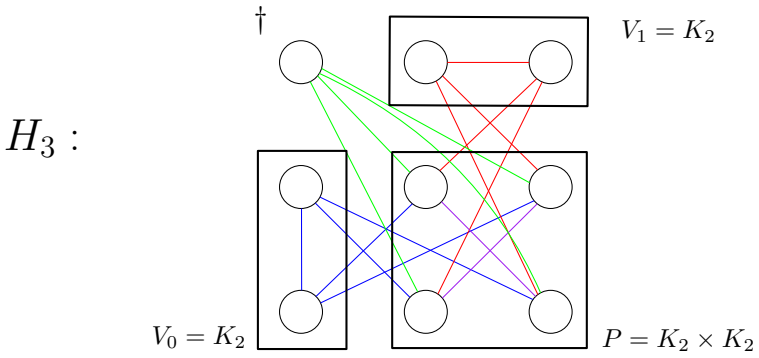

Figure 3: The maximal graph with the property Δ-(*) for $\Delta = 3$.

Remark 2. *It can be shown that for $\Delta = 3$ both the Chvátal and Grötsch graphs are suitable examples for $k = 4$.*

Remark 3. *As another application of his technique Marks showed in [86] that there is a Borel Δ-regular forest that does not admit Borel perfect matching. This holds even when we assume that the forest is Borel bipartite, i.e., it has Borel chromatic number 2. In order to show this result Marks works with free joins of colored hyperedges, that is, Cayley graphs of free products of cyclic groups. One should think of two types of triangles (3-hyperedges) that are joined in such a way that every vertex is contained in both types and there are no cycles. We remark that the playability condition can be formulated in this setting. Similarly, one can derive a version of Theorem 8. However, we do not have any application of this generalization.*

4.1 Applications of playability to homomorphism LCLs

In this section we find the graph examples from Theorem 9. First we introduce a condition Δ-(*) that is a weaker condition than having chromatic number at most Δ, but still implies that the homomorphism LCL is not playable. Then, we will show that–similarly to the way the complete graph on Δ-many vertices, K_Δ, is maximal among graphs of chromatic number $\leq \Delta$–there exists a maximal graph (under homorphisms) with property Δ-(*). Recall that we assume $\Delta > 2$.

Definition 12 (Property Δ-(*))**.** *Let $\Delta > 2$ and $G = (V, E)$ be a finite graph. We say that G satisfies property Δ-(*) if there are sets $S_0, S_1 \subseteq V$ such that G restricted to $V \setminus S_i$ has chromatic number at most $(\Delta - 1)$ for $i \in \{0, 1\}$, and there is no edge between S_0 and S_1.*

Note that $\chi(G) \leq \Delta$ implies Δ-(*): indeed, if $A_1, \ldots, A_\Delta$ are independent sets that cover $V(G)$, we can set $S_0 = S_1 = A_1$.

On the other hand, we claim that if G satisfies Δ-(*) then $\chi(G) \leq 2\Delta - 2$. In order to see this, take $S_0, S_1 \subseteq V(G)$ witnessing Δ-(*). Then, as there is no edge between S_0 and S_1, so in particular, between $S_0 \setminus S_1$ and $S_0 \cap S_1$, it follows that the chromatic number of G's restriction to S_0 is $\leq \Delta - 1$. But then we can construct proper $\Delta - 1$-colorings of S_0 and $V(G) \setminus S_0$, which shows our claim.

Proposition 1. *Let G be a graph satisfying Δ-(*). Then Π_G is not playable.*

Proof. Fix S_0, S_1 as in the definition of Δ-(*) and assume for a contradiction that colorings $\Lambda_1, \ldots, \Lambda_\Delta$ as described in Definition 11 exist. By Property Δ-(*), there exist independent sets $A_1, \ldots, A_{\Delta-1}$ such that S_0 together with the A_i covers G, i.e., such that $S_0 \cup A_1 \cup \cdots \cup A_{\Delta-1} = V(G)$.

For each $\alpha \in [\Delta - 1]$, we must have $\Lambda_\alpha(A_\alpha) =$ Alice, since otherwise condition (B) in Definition 11 is violated. Consequently $\Lambda_\Delta(S_0) =$ Bob, otherwise condition (A) is violated. Similarly $\Lambda_\Delta(S_1) =$ Bob. This shows that Λ_Δ does not satisfy condition (B) with $S = S_0$ and $T = S_1$. □

Next we describe maximal examples of graphs that satisfy the condition Δ-(*). That is, we define a graph H_Δ that satisfies Δ-(*), its chromatic number is $2\Delta - 2$ and every other graph that satisfies Δ-(*) admits a homomorphism in H_Δ. The graph H_3 is depicted in Figure 3.

Recall that the (categorical) product $G \times H$ of graphs G, H is the graph on $V(G) \times V(H)$, such that $((g,h),(g',h')) \in E(G \times H)$ iff $(g,g') \in E(G)$ and $(h,h') \in E(H)$.

Write P for the product $K_{\Delta-1} \times K_{\Delta-1}$. Let V_0 and V_1 be vertex disjoint copies of $K_{\Delta-1}$. We think of vertices in V_i and P as having labels from $[\Delta - 1]$ and $[\Delta - 1] \times [\Delta - 1]$, respectively. The graph H_Δ is the disjoint union of V_0, V_1, P and an extra vertex † that is connected by an edge to every vertex in P, and additionally, if v is a vertex in V_0 with label $i \in [\Delta - 1]$, then we connect it by an edge with $(i', j) \in P$ for every $i' \neq i$ and $j \in [\Delta - 1]$, and if v is a vertex in V_1 with label $j \in \Delta - 1$, then we connect it by an edge with $(i, j') \in P$ for every $j' \neq j$ and $i \in [\Delta - 1]$.

Proposition 2. *1. H_Δ satisfies Δ-(*).*

2. $\chi(H_\Delta) = 2\Delta - 2$.

3. A graph G satisfies Δ-(*) if and only if it admits a homomorphism to H_Δ.

Proof. (1) Set $S_0 = V(V_0) \cup \{\dagger\}$ and $S_1 = V(V_1) \cup \{\dagger\}$. By the definition there are no edges between S_0 and S_1. Consider now, e.g., $V(H_\Delta) \setminus S_0$. Let A_j consist of all elements in P that have second coordinate equal to j together with the vertex in V_1 that has the label j. By the definition, the set A_i is independent and $\bigcup_{i \in [\Delta-1]} A_i$ covers $H_\Delta \setminus S_0$, and similarly for S_1.

(2) By (1) and the claim after the definition of Δ-(*), it is enough to show that $\chi(H_\Delta) \geq 2\Delta - 2$. Towards a contradiction, assume that c is a proper coloring of H_Δ with $< 2\Delta - 2$-many colors. Note the vertex † guarantees that $|c(V(P))| \leq 2\Delta - 4$, and also $\Delta - 1 \leq |c(V(P))|$.

First we claim that there are no indices $i, j \in [\Delta - 1]$ (even with $i = j$) such that $c(i,r) \neq c(i,s)$ and $c(r,j) \neq c(s,j)$ for every $s \neq r$: indeed, otherwise, by the definition of P we had $c(i,r) \neq c(s,j)$ for every r, s unless $(i,r) = (s,j)$, which would the upper bound on the size of $c(V(P))$.

Therefore, without loss of generality, we may assume that for every $i \in [\Delta - 1]$ there is a color α_i and two indices $j_i \neq j_i'$ such that $c(i,j_i) = c(i,j_i') = \alpha_i$. It follows form the definition of P and $j_i \neq j_i'$ that $\alpha_i \neq \alpha_{i'}$ whenever $i \neq i'$.

Moreover, note that any vertex in V_1 is connected to at least one of the vertices (i,j_i) and (i,j_i'), hence none of the colors $\{\alpha_i\}_{i \in [\Delta-1]}$ can appear on V_1. Consequently, since V_1 is isomorphic to $K_{\Delta-1}$ we need to use at least $\Delta - 1$ additional colors, a contradiction.

(3) First note that if G admits a homomorphism into H_Δ, then the pullbacks of the sets witnessing Δ-(*) will witness that G has Δ-(*).

Conversely, let G be a graph that satisfies Δ-(*). Fix the corresponding sets S_0, S_1 together with $(\Delta - 1)$-colorings c_0, c_1 of their complements. We construct a homomorphism Θ from G to H_Δ. Let

$$\Theta(v) = \begin{cases} \dagger & \text{if } v \in S_0 \cap S_1, \\ c_0(v) & \text{if } v \in S_1 \setminus S_0, \\ c_1(v) & \text{if } v \in S_0 \setminus S_1, \\ (c_0(v), c_1(v)) & \text{if } v \notin S_0 \cup S_1. \end{cases}$$

Observe that $S = S_0 \cap S_1$ is an independent set such that there is no edge between S and $S_0 \cup S_1$. Using this observation, one easily checks case-by-case that Θ is indeed a homomorphism. □

Now, combining what we have so far, we can easily prove Theorem 9.

Proof of Theorem 9. It follows that Π_{H_Δ} is not playable from Proposition 2, Proposition 1. It is easy to see that if for a graph G the LCL Π_G is not playable then $\Pi_{G'}$ is not playable for every subgraph G' of G. Since erasing a vertex decreases the chromatic number by at most one, for each $k \leq 2\Delta - 2$ there is a subgraph G_k of H_Δ with $\chi(G_k) = k$, such that Π_{G_k} is not playable.

It follows from Theorems 7 and 8 that there is a Δ-regular Borel forest that admits no Borel homomorphism to any graph of G_k and that $\Pi_{G_k} \notin \mathsf{LOCAL}(O(\log^*(n)))$.

Finally, note that if $k \geq \Delta$ then G_k can be chosen so that it contains K_Δ, yielding $\Pi_{G_k} \in \mathsf{RLOCAL}(O(\log\log(n))$. □

Remark 4. *Hedetniemi's conjecture is the statement that if G, H are finite graphs then $\chi(G \times H) = \min\{\chi(G), \chi(H)\}$. This conjecture has been recently disproven by Shitov [100], and strong counterexamples have been constructed later (see, [103, 109]). We claim that these imply for $\varepsilon > 0$ the existence of finite graphs H with $\chi(H) \geq (2-\varepsilon)\Delta$ to which Δ-regular Borel forests cannot have a homomorphism in BOREL, for every large enough Δ. Indeed, if a Δ-regular Borel forest admitted a Borel homomorphism to each finite graph of chromatic number at least $(2-\varepsilon)\Delta$, it would have such a homomorphism to their product as well. Thus, we would obtain that the chromatic number of the product of any graphs of chromatic number $(2-\varepsilon)\Delta$ is at least $\Delta+1$. This contradicts Zhu's result [109], which states that the chromatic number of the product of graphs with chromatic number n can drop to $\approx \frac{n}{2}$.*

Remark 5. *A natural attempt to construct graphs with large girth and not playable homomorphisms problem would be to consider random d-regular graphs of size n for a large enough n. However, it is not hard to see that setting $\Lambda(A) =$ Alice if and only if $|A| < \frac{n}{d}$ shows that this approach cannot work.*

4.2 Proof of Theorem 7

In this section we prove Theorem 7, by applying Marks' game technique in the LOCAL setting. In order to define our games, we will need certain auxiliary graphs, the so-called *ID graphs*. The purpose of these graphs is to define a "playground" for the games that we consider. Namely, vertices in the game are labeled by vertices from the ID graph in such a way that the at the end we obtain a homomorphism from our underlying graph to the ID graph.

Definition 13. *Let $n, t \in \mathbb{N}$ and $r \in \mathbb{R}^+$. A pair $\mathbf{H}_{n,t,r} = (H_{n,t,r}, c)$ is called an* ID graph*, if*

1. *$H_{n,t,r}$ is graph with girth at least $2t+2$,*
2. *$|V(H_{n,t,r})| \leq n$,*
3. *c is a Δ-edge-coloring of $H_{n,t,r}$, such that every vertex is adjacent to at least one edge of each color,*
4. *for each $\alpha \in [\Delta]$ and $H^\alpha_{n,t,r} = (V(H_{n,t,r}), E(H_{n,t,r}) \cap c^{-1}(\alpha))$ (i.e., $H^\alpha_{n,t,r}$ is the graph formed by α-colored edges) we have that the independence ratio of $H^\alpha_{n,t,r}$ is at most r.*

Before we define the game we show that ID graphs exist.

Proposition 3. *Let $t_n \in o(\log(n))$, $r > 0$, $\Delta \geq 2$. Then there is an ID graph $\mathbf{H}_{n,t_n,r}$ for every $n \in \mathbb{N}$ sufficiently large.*

Proof. We use the *configuration model* for regular random graphs, see [108]. This model is defined as follows. Let n be even. Then a d-regular random sample on n-vertices is just a union of d-many independent uniform random perfect matchings. Note that in this model we allow parallel edges.

It was proved by Bollobás [23] that the independence ratio of a random d-regular graph is at most $\frac{2\log(d)}{d}$ a.a.s. Moreover, this quantity is concentrated by an easy application of McDiarmid's result [89], i.e.,

$$\mathrm{P}\left(|X - \mathbf{E}(X)| \geq \sqrt{n}\right) < 2\exp\left(-\frac{n}{d}\right), \tag{1}$$

where X is the random variable that counts the size of a maximal independent set. Therefore for fixed n large enough we have that the independence ratio of a random sample is at most $\frac{3\log(d)}{d}$ with probability at least $\left(1 - 2\exp\left(-\frac{n}{d}\right)\right)$.

Pick a d large enough such that $\frac{3\log(d)}{d} < r$. Now for an n large enough take Δ-many independent samples of random d-regular graphs according to the configuration model. Note that this is a random sample from the configuration model for Δd-regular graphs. We define c to be equal to $\alpha \in [\Delta]$ on edges of the α-th sample. Then condition (4) is satisfied with probability at least

$$\left(1 - 2\exp\left(-\frac{n}{d}\right)\right)^{\Delta}.$$

It remains to show that the girth condition is satisfied. Recall that we assume $t_n \in o(\log n)$. Then we have $(\Delta d - 1)^{2t_n - 1} \in o(n)$. Using [90, Corollary 1] we have that the probability of having girth at least t_n is

$$\begin{aligned} \exp\left(-\sum_{a=3}^{t_n} \frac{(\Delta d - 1)^a}{2a} + o(1)\right) &\geq \exp\left(-(\Delta d)^{t_n}\right) \\ &\geq \exp(-o(n)) \\ &> 1 - \left(1 - 2\exp\left(-\frac{n}{d}\right)\right)^{\Delta} \end{aligned} \tag{2}$$

as $n \to \infty$. This shows that there exists such a graph $\mathbf{H}_{t_n,n,r}$ with non-zero probability. □

Next, we define the games. As mentioned before, the games are going to depend on the following parameters: an algorithm $\mathcal{A}_n$ of local complexity $t \in o(\log(n))$, an ID graph $\mathbf{H}_{n,t,r}$, $\alpha \in \Delta$, $\sigma \in V(H_{n,t,r})$ and $S \subseteq \Sigma$. (We will view $\mathcal{A}_n$, $\mathbf{H}_{n,t,r}$ as fixed, and the rest of the parameters as variables).

The game

$$\mathbb{G}(\mathcal{A}_n, n, t, \mathbf{H}_{n,t,r})[\alpha, \sigma, S]$$

is defined as follows: two players, Alice and Bob assign labels to the vertices of a rooted Δ-regular tree of diameter t. The labels are vertices of $H_{n,t,r}$ and the root is labeled σ. In the k-th round, where $0 < k \leq t$, first Alice labels vertices of distance k from the root on the side of the α edge. After that, Bob labels all remaining vertices of distance k, etc (see Figure 4). We also require the assignment of labels to give rise to an edge-color preserving homomorphism to $\mathbf{H}_{n,t,r}$. (For example, if it is Alice's turn to label a neighbor of some vertex v, that has been assigned a label $\rho \in V(H_{n,t,r})$ in the previous round, along an edge that has color β, then the allowed labels are only those that span a β edge with ρ in $H_{n,t,r}$).

By property (2) of the ID graph, we can fix an injective map from $V(H_{n,t,r})$ to $[n]$. Now, we say that Alice wins an instance of the game iff $\mathcal{A}_n$ applied to the produced labeling of the rooted tree does **not** produce an element of S on the half edge that starts in the root and has edge color

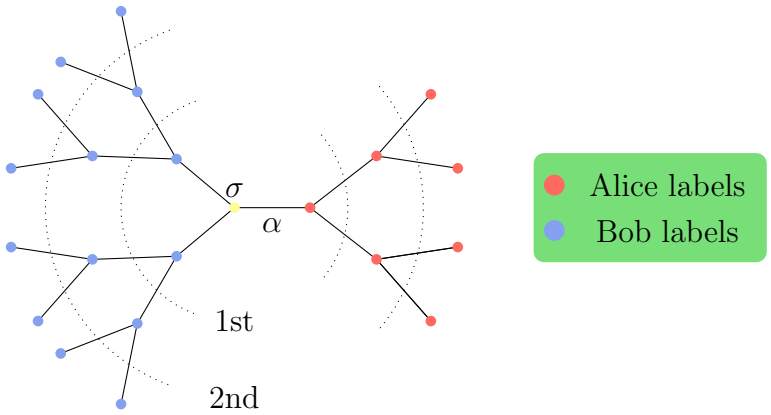

Figure 4: The game $\mathbb{G}(\mathcal{A}_n, n, t, \mathbf{H}_{n,t,r})[\alpha, \sigma, S]$

α. Note that this is well defined thanks to our assumption on the girth of $H_{n,t,r}$. Let us define $\Lambda^\sigma_\alpha(S)$ to be Alice or Bob depending on who has a winning strategy in the game

$$\mathbb{G}(\mathcal{A}_n, n, t, \mathbf{H}_{n,t,r})[\alpha, \sigma, S].$$

Since the game is finite and there is no draw, one of the players has a winning strategy. Thus, the map

$$\Lambda^\sigma_\alpha : \mathbb{P}_\alpha \to \{\text{Alice}, \text{Bob}\}$$

is well-defined for all $\sigma \in V(H_{n,t,r})$ and $\alpha \in [\Delta]$.

Proposition 4. *Let $\mathcal{A}_n$ be an algorithm of local complexity $t \in o(\log(n))$ that solves Π and $\sigma \in V(H_{n,t,r})$. Then $(\Lambda^\sigma_\alpha)_{\alpha\in\Delta}$ satisfies (A) in Definition 11.*

Proof. Assume that $(S_\alpha)_{\alpha\in\Delta}$ is such that $\Lambda^\sigma_\alpha(S_\alpha) = \text{Alice}$. This means that Alice has winning strategy in all the games corresponding to S_α. Letting these strategies play against each other in the obvious way, produces a labeling of the tree. Since $\mathcal{A}_n$ solves Π it has to output labeling of half edges $(\mathtt{a}_\alpha)_{\alpha\in\Delta} \in \mathcal{N}$, where $\mathtt{a}_\alpha$ is a label on the half edge that start at the root and has color α. Note that we must have $\mathtt{a}_\alpha \notin S_\alpha$ by the definition of winning strategy for Alice. This shows that (A) of Definition 11 holds. □

We are ready to prove Theorem 7.

Proof of Theorem 7. Let $(\mathcal{A}_n)_{n\in\mathbb{N}}$ be a sequence of algorithms of local complexity $t_n \in o(\log(n))$ that solve Π and $N \in \mathbb{N}$ be the number of all possible colorings of vertices of $(\mathbb{P}_\alpha)_{\alpha\in\Delta}$ with two colors, that is, $N = 2^{\sum_\alpha |V(\mathbb{P}_\alpha)|}$. Set $r := \frac{1}{N+1}$. By Proposition 3 there exists an ID graph $\mathbf{H}_{n,t,r}$. Thus, the games above are well-defined and we can construct the functions $(\Lambda^\sigma_\alpha)_{\alpha\in\Delta}$. Since there are only N possibilities for such a sequence, there exists a set $X \subseteq V(H_{n,t,r})$ of relative size greater than r, such that for all $\sigma \in X$ the sequence of functions $(\Lambda^\sigma_\alpha)_{\alpha\in\Delta}$ is the same.

Since Π is not playable, we can find an edge color $\alpha \in \Delta$ and sets $S, T \in \mathbb{P}_\alpha$ such that Λ^σ_α does not satisfy (B) from Definition 11 with S, T for every $\sigma \in X$. Note that this is because (A) is always satisfied by Proposition 4.

By property (4) of the ID graph, there exist $\sigma_0, \sigma_1 \in X$ that span an α edge in $H_{n,t,r}$. We let the winning strategies of Bob in the games

$$\mathbb{G}(\mathcal{A}_n, n, t, \mathbf{H}_{n,t,r})[\alpha, \sigma_0, S],\ \mathbb{G}(\mathcal{A}_n, n, t, \mathbf{H}_{n,t,r})[\alpha, \sigma_1, T],$$

play against each other, where we start with an α edge with endpoints labeled by σ_0, σ_1. This produces a labeling of the vertices that have distance at most t from either of the endpoints of the edge (intuitively, the tree "rooted" at this edge of "diameter" $t + \frac{1}{2}$). Now applying $\mathcal{A}_n$ on the vertex with label σ_0 produces $\mathtt{a}_0 \in S$ on the the half edge that starts at this vertex and has color α. Similarly, we produce $\mathtt{a}_1 \in T$. However, $(\mathtt{a}_0, \mathtt{a}_1) \notin \mathbf{E}_\alpha$ by the definition of an edge in $\mathbb{P}_\alpha$. This shows that $\mathcal{A}_n$ does not solve Π. □

4.3 Proof of Theorem 8

We show how to use an infinite analogue of the ID graph to prove our main result in the Borel context, Theorem 8. Finding such a graph/graphing is in fact much easier in this context.

Definition 14. *Let $r \in \mathbb{R}^+$. $\mathbf{H}_r = (\mathcal{H}_r, c)$ is an* ID graphing*, if*

1. *$\mathcal{H}_r$ is an acyclic locally finite Borel graphing on a standard probability measure space (X, μ),*
2. *c is a Borel Δ-edge-coloring of $\mathcal{H}_r$, such that every vertex is adjacent to at least one edge of each color,*
3. *for each $\alpha \in [\Delta]$ the μ-measure of a maximal independent set of $\mathcal{H}_r^\alpha = (V(\mathcal{H}_r), E(\mathcal{H}_r) \cap c^{-1}(\alpha))$ is at most r.*

Proposition 5. *For each $r > 0$ there exists an ID graphing $\mathbf{H}_r$.*

Proof. Let $\mathcal{H}_r$ be a *local-global limit* of the random graphs constructed in Proposition 3 (see, e.g., [65] for the basic results about local-global convergence). It is not hard to check that this limit satisfies the required properties. □

Now we turn to the proof of Theorem 8. The proof will closely follow the argument given in the proof of the LOCAL version, i.e., Theorem 7, but can be understood without reading the latter.

Let Π be an LCL that is not playable, and $N \in \mathbb{N}$ be the number of all possible colorings of vertices of $(\mathbb{P}_\alpha)_{\alpha \in \Delta}$ with two colors, that is, $N = 2^{\sum_\alpha |V(\mathbb{P}_\alpha)|}$. Set $r := \frac{1}{N+1}$.

We define a Borel acyclic Δ-regular graph $\mathcal{G}$, with edges properly colored by Δ, that does not admit a Borel solution of Π. Vertices of $\mathcal{G}$ are pairs (x, A), where $x \in X$ is a vertex of $\mathcal{H}_r$ and A is a countable subgraph of $\mathcal{H}_r$ that is a Δ-regular tree that contains x and the edge coloring of $\mathbf{H}_r$ induces a proper edge coloring of A. We say that (x, A) and (y, B) are connected by an α-edge in $\mathcal{G}$ if $A = B$, x, y are adjacent in A and the edge that connects them has color $\alpha \in \Delta$.

Suppose for a contradiction that $\mathcal{A}$ is a Borel function that solves Π on $\mathcal{G}$.

Next, we define a family of games parametrized by $\alpha \in [\Delta]$, $x \in V(\mathcal{H}_r)$ and $S \subseteq \Sigma$. For the reader familiar with Marks' construction, let us point out that for a fixed x, the games are analogues to the ones he defines, with the following differences: allowed moves are vertices of the ID graphing $\mathcal{H}_r$ and restricted by its edge relation, and the winning condition is defined by a set of labels, not just merely one label.

So, the game

$$\mathbb{G}(\mathcal{A}, \mathbf{H}_r)[\alpha, x, S]$$

is defined as follows: Alice and Bob alternatingly label vertices of a Δ-regular rooted tree. The root is labelled by x, and the labels come from $V(\mathcal{H}_r)$. In the k-th round, first Alice labels vertices of distance k from the root on the side of the α edge. After that, Bob labels all remaining vertices of distance k, etc (see Figure 4). We also require the assignment of labels to give rise to an edge-color preserving homomorphism to $\mathbf{H}_r$.

It follows from the acyclicity of $\mathcal{H}_r$ that a play of a game determines a Δ-regular rooted subtree of $\mathcal{H}_r$ to which the restriction of the edge-coloring is proper. That is, it determines a vertex (x, A) of $\mathcal{G}$. Let Alice win iff the output of $\mathcal{A}$ on the half-edge determined by (x, A) and the color α is **not** in S.

Define the function $\Lambda_\alpha^x : \mathbb{P}_\alpha \to \{\text{Alice}, \text{Bob}\}$ assigning the player to some $S \in V(\mathbb{P}_\alpha)$ who has a winning strategy in $\mathbb{G}(\mathcal{A}, \mathbf{H}_r)[\alpha, x, S]$. Note that, since $\mathcal{H}_r$ is locally finite, each player has only finitely many choices at each position. Thus, it follows from Borel Determinacy Theorem that Λ_α^x is well defined.

Now we show the analogue of Proposition 4.

Proposition 6. *$(\Lambda_\alpha^x)_{\alpha\in\Delta}$ satisfies (A) in Definition 11.*

Proof. Assume that $(S_\alpha)_{\alpha\in\Delta}$ is such that $\Lambda_\alpha^x(S_\alpha) = \text{Alice}$. This means that Alice has winning strategy in all the games corresponding to S_α. Letting these strategies play against each other in the obvious way, produces a vertex (x, A) of $\mathcal{G}$. Since $\mathcal{A}$ solves Π it has to output labeling of half edges $(\mathtt{a}_\alpha)_{\alpha\in\Delta} \in \mathcal{N}$, where $\mathtt{a}_\alpha$ is a label on the half edge that start at (x, A) and has color α. Note that we must have $\mathtt{a}_\alpha \notin S_\alpha$ by the definition of winning strategy for Alice. This shows that (A) of Definition 11 holds. □

Now we are ready to prove Theorem 8.

Proof of Theorem 8. Let f be defined by

$$x \mapsto (\Lambda_\alpha^x)_{\alpha\in\Delta}.$$

Note that f has a finite range. Using the fact that the allowed moves for each player can be determined in a Borel way, uniformly in x, it is not hard to see that for each element s in the range, $f^{-1}(s)$ is in the algebra generated by sets that can be obtained by applying game quantifiers to Borel sets (for the definition see [75, Section 20.D]). It has been shown by Solovay [101] and independently by Fenstad-Norman [48] that such sets are provably Δ_2^1, and consequently, measurable. Therefore, f is a measurable map.

As the number of sequences of functions $(\Lambda_\alpha^x)_{\alpha\in\Delta}$ is $\leq N$, by the choice of r, there exists a Borel set Y with $\mu(Y) > r$, such that f is constant on Y.

Since Π is not playable Proposition 6 gives that there is an $\alpha \in \Delta$ and $S, T \in \mathbb{P}_\alpha$ that violate condition (B). Recall that the measure of an independent Borel set of $\mathcal{H}_r^\alpha$ is at most r. That means that there are $x, y \in Y$ such that (x, y) is an α-edge in $\mathcal{H}_r$. We let the winning strategies of Bob in the games

$$\mathbb{G}(\mathcal{A}, \mathbf{H}_r)[\alpha, x, S],\ \mathbb{G}(\mathcal{A}, \mathbf{H}_r)[\alpha, y, T],$$

play against each other, where we start with an α edge with endpoints labeled by x, y. This produces a labeling of a Δ-regular tree A with labels from $\mathcal{H}_r$ such that (x, A) and (y, A) span an α-edge in $\mathcal{H}_r$. Applying $\mathcal{A}$ on the half-edge determined by (x, A) and the color α, we obtain $\mathtt{a}_0 \in S$. Similarly, we produce $\mathtt{a}_1 \in T$ for (y, A). However, $(\mathtt{a}_0, \mathtt{a}_1) \notin \mathbf{E}_\alpha$ by the definition of an edge in $\mathbb{P}_\alpha$. This shows that $\mathcal{A}$ does not produce a Borel solution to Π. □

5 Separation Of Various Complexity Classes

In this section we provide examples of several problems that separate some classes from Figure 1. The examples show two things. First, we have $\mathsf{OLOCAL}(\text{poly}\log 1/\varepsilon) \neq \mathsf{OLOCAL}(O(\log\log 1/\varepsilon))$. This shows that there are problems such that their worst case complexity is at least $\Theta(\log n)$ on

finite Δ-regular trees, but their average local complexity is constant. Second, we show that there are problems in the class BOREL that are not in the class $\mathsf{RLOCAL}(O(\log\log n)) = \mathsf{RLOCAL}(o(\log n))$, on Δ-regular trees. This shows that one cannot in general hope that results from (Borel) measurable combinatorics can be turned into very efficient (sublogarithmic) distributed algorithms.

5.1 Preliminaries

We first show that there is a strong connection between randomized local complexities and uniform local complexities. Afterwards, we introduce a generic construction that turns any LCL into a new LCL. We later use this generic transformation to construct an LCL that is contained in the set $\mathsf{BOREL} \setminus \mathsf{RLOCAL}(O(\log\log n))$.

Uniform vs Local Randomized Complexity We will now discuss the connections between uniform and randomized complexities. Note that the easy part of the connection between the two concepts is turning uniform local algorithms into randomized ones, as formalized in the following proposition.

Proposition 7. *We have* $\mathsf{ULOCAL}(t(\varepsilon)) \subseteq \mathsf{RLOCAL}(t(1/n^{O(1)}))$.

Proof. We claim that a uniform local algorithm $\mathcal{A}$ with a uniform local complexity of $t(\varepsilon)$ can be turned into a local randomized algorithm $\mathcal{A}'$ with a local complexity of $t(1/n^{O(1)})$.

The algorithm $\mathcal{A}'$ simulates $\mathcal{A}$ on an infinite Δ-regular tree – each vertex u of degree less than Δ in the original tree pretends that the tree continues past its virtual half-edges and the random bits of u are used to simulate the random bits in this virtual subtree. Choosing $\varepsilon = 1/n^{O(1)}$, one gets that the probability of $\mathcal{A}'$ needing to look further than $t(\varepsilon)$ for any vertex is bounded by $1/n^{O(1)}$, as needed in the definition of the randomized local complexity. □

On the other hand, we will use the following proposition from [62]. It informally states that the existence of *any* uniform local algorithm together with the existence of a sufficiently fast randomized local algorithm for a given LCL Π directly implies an upper bound on the uniform complexity of Π.

Proposition 8 ([62])**.** *Let $\mathcal{A}$ be a uniform local algorithm solving an LCL problem Π such that its* randomized *local complexity on finite Δ-regular trees is $t(n)$ for $t(n) = o(\log n)$. Then, the uniform local complexity of $\mathcal{A}$ on infinite Δ-regular trees is $O(t(1/\varepsilon))$.*

Proposition 8 makes our life simpler, since we can take a known randomized distributed local algorithm with local complexity $g(n) = o(\log n)$ and check if it works without the knowledge of n. If yes, this automatically implies that the uniform local complexity of the algorithm is $h(\varepsilon) = O(g(1/\varepsilon))$. In particular, combining Proposition 8 with the work of [34, 35] and verifying that the algorithms of Fischer and Ghaffari [49, Section 3.1.1] and Chang et al. [33, Section 5.1] work without the knowledge of n, we obtain the following result.

Theorem 10. *We have:*

- $\mathsf{LOCAL}(O(1)) = \mathsf{OLOCAL}(O(1))$
- $\mathsf{RLOCAL}(O(\log^* n)) = \mathsf{OLOCAL}(O(\log^* 1/\varepsilon))$
- $\mathsf{RLOCAL}(O(\log\log n)) = \mathsf{OLOCAL}(O(\log\log 1/\varepsilon))$

Moreover, there are no other possible uniform local complexities for $t(\varepsilon) = o(\log 1/\varepsilon)$.

We note that the first two items are proven in [62]. For the third item it suffices by known reductions to find a uniform algorithm solving a version of the distributed Lovász Local Lemma (LLL) on so-called tree-structured dependency graphs considered in [33].

Proof sketch. The proof follows from Proposition 7 and the following ideas. The first item follows from the fact that any local algorithm with local complexity $O(1)$ can simply be made uniform. More specifically, there exists a constant n_0 — depending only on the problem Π — such that the algorithm, being told the size of the graph is n_0, is correct on *any* graph of size $n \geq n_0$.

For the second item, by the work of [34, 35], it suffices to check that there is a uniform distributed $(\Delta+1)$-coloring algorithm with uniform local complexity $O(\log^* 1/\varepsilon)$. Such an algorithm was given in [72], or follows from the work in [78] and Proposition 8.

Similarly, for the third item, by the work of [35] it suffices to check that there is a uniform distributed algorithm for a specific LLL problem on trees with uniform local complexity $O(\log\log 1/\varepsilon)$. Such an algorithm can be obtained by combining the randomized *pre-shattering* algorithm of Fischer and Ghaffari [49, Section 3.1.1] and the deterministic *post-shattering* algorithm of Chang et al. [33, Section 5.1] in a *graph shattering* framework [16], which solves the LLL problem with local complexity $O(\log\log n)$. By Proposition 8, it suffices to check that this algorithm can be made to work even if it does not know the size of the graph n. We defer the details to Appendix B. This finishes the proof of Theorem 10. □

Adding Paths Before we proceed to show some separation results, we define a certain construction that turns any LCL problem Π into a new LCL problem $\overline{\Pi}$ with the following property. If the original problem Π cannot be solved by a fast local algorithm, then the same holds for $\overline{\Pi}$. However, $\overline{\Pi}$ might be strictly easier to solve than Π for BOREL constructions.

Definition 15. *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL. We define an LCL $\overline{\Pi} = (\Sigma', \mathcal{V}', \mathcal{E}')$ as follows. Let Σ' be Σ together with one new label. Let $\mathcal{V}'$ be the union of $\mathcal{V}$ together with any cardinality-Δ multiset that contains the new label exactly two times. Let $\mathcal{E}'$ be the union of $\mathcal{E}$ together with the cardinality-2 multiset that contains the new label twice.*

In other words, the new label determines doubly infinite lines in infinite Δ-regular trees, or lines that start and end in virtual half-edges in finite Δ-regular trees. Moreover, a vertex that is on such a line does not have to satisfy any other vertex constraint. We call these vertices *line-vertices* and each edge on a line a *line-edge*.

Proposition 9. *Let Π be an LCL problem such that $\overline{\Pi} \in \mathsf{RLOCAL}(t(n))$ for $t(n) = o(\log(n))$. Then also $\Pi \in \mathsf{RLOCAL}(O(t(n)))$.*

Proof. Let $\overline{\mathcal{A}}$ be a randomized LOCAL algorithm that solves $\overline{\Pi}$ in $t(n) = o(\log n)$ rounds with probability at least $1 - 1/n^C$ for some sufficiently large constant C. We will construct an algorithm $\mathcal{A}$ for Π with complexity $t(n)$ that is correct with probability $1 - \frac{4}{n^{C/3-2}}$. The success probability of $\mathcal{A}$ can then be boosted by "lying to it" that the number of vertices is $n^{O(1)}$ instead of n; this increases the running time by at most a constant factor and boosts the success probability back to $1 - 1/n^C$.

Consider a Δ-regular rooted finite tree T of depth $10t(n)$ and let u be its root vertex. Note that $|T| \leq \Delta^{10t(n)+1} < n$, for n large enough.

We start by proving that when running $\overline{\mathcal{A}}$ on the tree T, then u is most likely not a line-vertex. This observation then allows us to turn $\overline{\mathcal{A}}$ into an algorithm $\mathcal{A}$ that solves Π on any Δ-regular input tree. Let X be the indicator of $\mathcal{A}$ marking u as a line-vertex. Moreover, for $i \in [\Delta]$, let Y_i

be the indicator variable for the following event. The number of line edges in the i-th subtree of u with one endpoint at depth $5t(n)$ and the other endpoint at depth $5t(n)+1$ is odd.

By a simple parity argument we can relate the value of X with the values of $Y_1, Y_2, \ldots, Y_\Delta$ as follows. If $X = 0$, that is, u is not a line-vertex, then all of the Y_i's have to be 0, as each path in the tree T is completely contained in one of the Δ subtrees of u. On the other hand, if u is a line-vertex, then there exists exactly one path, the one containing u, that is not completely contained in one of the Δ subtrees of u. This in turn implies that exactly two of the Δ variables $Y_1, Y_2, \ldots, Y_\Delta$ are equal to 1.

The random variables $Y_1, Y_2, \ldots, Y_\Delta$ are identically distributed and mutually independent. Hence, if $\mathrm{P}(Y_i = 1) > \frac{1}{n^{C/3}}$, then the probability that there are at least 3 Y_i's equal to 1 is strictly greater than $\left(\frac{1}{n^{C/3}}\right)^3 = \frac{1}{n^C}$. This is a contradiction, as in that case $\overline{\mathcal{A}}$ does not produce a valid output, which according to our assumption happens with probability at most $\frac{1}{n^C}$.

Thus, we can conclude that $\mathrm{P}(Y_i = 1) \leq \frac{1}{n^{C/3}}$. By a union bound, this implies that all of the Y_i's are zero with probability at least $\frac{1}{n^{C/3-1}}$, and in that case u is not a line-vertex.

Finally, the algorithm $\mathcal{A}$ simulates $\overline{\mathcal{A}}$ as if the neighborhood of each vertex was a Δ-regular branching tree up to depth at least $t(n)$.

It remains to analyze the probability that $\mathcal{A}$ produces a valid solution for the LCL problem Π. To that end, let v denote an arbitrary vertex of the input tree. The probability that the output of v's half edges satisfy the vertex constraint of the LCL problem $\overline{\Pi}$ is at least $1 - 1/n^C$. Moreover, in the case that v is not a line-vertex, which happens with probability at least $1 - \frac{1}{n^{C/3-1}}$, the output of v's half edges even satisfy the vertex constraint of the LCL problem Π. Hence, by a union bound the vertex constraint of the LCL problem Π around v is satisfied with probability at least $1 - \frac{2}{n^{C/3-1}}$. With exactly the same reasoning, one can argue that the probability that the output of $\mathcal{A}$ satisfies the edge constraint of the LCL problem Π at a given edge is also at least $1 - \frac{2}{n^{C/3-1}}$. Finally, doing a union bound over the n vertex constraints and $n-1$ edge constraints it follows that $\mathcal{A}$ produces a valid solution for Π with probability at least $1 - \frac{4}{n^{C/3-2}}$. □

Remark 6. *The ultimate way how to solve LCLs of the form $\overline{\Pi}$ is to find a spanning forest of lines. This is because once we have a spanning forest of lines, then we might declare every vertex to be a line-vertex and label every half-edge that is contained on a line with the new symbol in $\overline{\Pi}$. The remaining half-edges can be labeled arbitrarily in such a way that the edge constraints are satisfied.*

Put otherwise, once we are able to construct a spanning forest of lines and $\mathcal{E} \neq \emptyset$, where $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$, then $\overline{\Pi}$ can be solved. Moreover, in that case the complexity of $\overline{\Pi}$ is upper bounded by the complexity of finding the spanning forest of lines.

Lyons and Nazarov [83] showed that $\Pi_{pm} \in$ fiid and it is discussed in [82] that the construction can be iterated $(\Delta-2)$-many times. After removing each edge that is contained in one of the $\Delta-2$ perfect matchings each vertex has a degree of two. Hence, it is possible to construct a spanning forest of lines as fiid. Consequently, $\overline{\Pi} \in$ fiid for every $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ with $\mathcal{E} \neq \emptyset$.

5.2 LOCAL$(O(\log^* n)) \neq$ BOREL

We now formalize the proof that LOCAL$(O(\log^* n)) \neq$ BOREL from Section 2.2. Recall that $\Pi_{\chi,\Delta}$ is the proper vertex Δ-coloring problem. The following claim directly follows by Proposition 9 together with the fact that $\Pi_{\chi,\Delta} \notin$ LOCAL$(O(\log^* n))$.

Claim 2. *We have $\overline{\Pi_{\chi,\Delta}} \notin$ LOCAL$(O(\log^* n))$.*

On the other hand we show that adding lines helps to find a Borel solution. This already provides a simple example of a problem in the set $\mathsf{BOREL} \setminus \mathsf{LOCAL}(O(\log^* n))$.

Proposition 10. *We have* $\overline{\Pi_{\chi,\Delta}} \in \mathsf{BOREL}$.

Proof. By [77], every Borel graph of finite maximum degree admits a Borel maximal independent set. Let $\mathcal{G}$ be a Borel Δ-regular acyclic graph on a standard Borel space X. By an iterative application of the fact above we find Borel sets $A_1, \dots, A_{\Delta-2}$ such that A_1 is a maximal independent set in $\mathcal{G}$ and A_i is a maximal independent set in $\mathcal{G} \setminus \bigcup_{j<i} A_j$ for every $i > 1$. We think of $A_1, \dots, A_{\Delta-2}$ as color classes for the first $\Delta - 2$ colors. Let $B = X \setminus \bigcup_{i \in [\Delta-2]} A_i$ and let $\mathcal{H}$ be the subgraph of $\mathcal{G}$ determined by B. It is easy to see that the maximum degree of $\mathcal{H}$ is 2. In particular, $\mathcal{H}$ consists of finite paths, one-ended infinite paths or doubly infinite paths. We use the extra label in $\overline{\Pi_{\chi,\Delta}}$ to mark the doubly infinite paths. It remains to use the remaining 2 colors in $[\Delta]$ to define a proper vertex 2-coloring of the finite and one-ended paths. It is a standard argument that this can be done in a Borel way and it is easy to verify that, altogether, we found a Borel $\overline{\Pi_{\chi,\Delta}}$-coloring of $\mathcal{G}$. □

5.3 Examples and Lower Bound

In this subsection we define two LCLs and show that they are not in the class $\mathsf{RLOCAL}(o(\log n))$. In the next subsection we show that one of them is in the class $\mathsf{OLOCAL}(\operatorname{poly}\log 1/\varepsilon)$ and the other in the class BOREL. Both examples that we present are based on the following relaxation of the perfect matching problem.

Definition 16 (Perfect matching in power-2 graph)**.** *Let* Π^2_{pm} *be the perfect matching problem in the power-*2 *graph, i.e., in the graph that we obtain from the input graph by adding an edge between any two vertices that have distance at most* 2 *in the input graph.*

We show that Π^2_{pm} is not contained in $\mathsf{RLOCAL}(o(\log n))$. Proposition 9 then directly implies that $\overline{\Pi^2_{\mathrm{pm}}}$ is not contained in $\mathsf{RLOCAL}(o(\log n))$ as well. On the other hand, we later use a one-ended spanning forest decomposition to show that Π^2_{pm} is in $\mathsf{OLOCAL}(\operatorname{poly}\log 1/\varepsilon)$ and a one or two-ended spanning forest decomposition to show that $\overline{\Pi^2_{\mathrm{pm}}}$ is in BOREL.

We first show the lower bound result. The proof is based on a simple parity argument as in the proof of Proposition 9.

Theorem 11. *The problems* Π^2_{pm} *and* $\overline{\Pi^2_{pm}}$ *are not in the class* $\mathsf{RLOCAL}(o(\log n))$.

Proof. Suppose that the theorem statement does not hold for Π^2_{pm}. Then, by Theorem 5, there is a distributed randomized algorithm solving Π^2_{pm} with probability at least $1 - 1/n$. By telling the algorithm that the size of the input graph is $n^{2\Delta}$ instead of n, we can further boost the success probability to $1 - 1/n^{2\Delta}$. The resulting round complexity is at most a constant factor larger, as $\Delta = O(1)$. We can furthermore assume that the resulting algorithm $\mathcal{A}$ will always output for each vertex v a vertex $M(v) \neq v$ in v's 2-hop neighborhood, even if $\mathcal{A}$ fails to produce a valid solution. The vertex $M(v)$ is the vertex v decides to be matched to, and if $\mathcal{A}$ produces a valid solution it holds that $M(M(v)) = v$.

Consider a fully branching Δ-regular tree T of depth $10t(n)$. Note that $|T| < n$ for n large enough. Let u be the root of T and T_i denote the i-th subtree of u (so that $u \notin T_i$). Let S_i denote the set of vertices v in T_i such that both v and $M(v)$ have distance at most $2t(n)$ from u. Note that $M(v)$ is not necessarily a vertex of T_i. Let Y_i be the indicator of whether S_i is even.

Observe that, if $\mathcal{A}$ does not fail on the given input graph, then the definition of the S_i implies that every vertex in $\{u\} \cup \bigcup_{i \in [\Delta]} S_i$ is matched to a vertex in $\{u\} \cup \bigcup_{i \in [\Delta]} S_i$. Hence, the number

of vertices in $\{u\} \cup \bigcup_{i\in[\Delta]} S_i$ is even, which implies that we cannot have $Y_1 = Y_2 = \ldots = Y_\Delta = 1$ unless $\mathcal{A}$ fails. Note that Y_i depends only on the output of $\mathcal{A}$ at the vertices in T_i that have a distance of precisely $2t(n) - 1$ or $2t(n)$ to u (as all other vertices in T_i are guaranteed to be in S_i). Hence, the events $Y_i = 1$, $i \in [\Delta]$, are independent, and, since $\mathrm{P}(Y_1 = 1) = \ldots = \mathrm{P}(Y_\Delta = 1)$ (as all vertices in all T_i see the same topology in their $t(n)$-hop view), we obtain $(\mathrm{P}(Y_1 = 1))^\Delta \leq 1/n^{2\Delta}$, which implies $\mathrm{P}(Y_i = 1) \leq 1/n^2$, for any $i \in [\Delta]$.

Hence, with probability at least $1 - \Delta/n^2$ we have $Y_1 = Y_2 = \ldots = Y_\Delta = 0$, by a union bound. Let $v_1, v_2, \ldots, v_\Delta$ denote the neighbors of u with v_i being the root of subtree T_i. Note that if $\mathcal{A}$ does not fail and $Y_1 = Y_2 = \ldots = Y_\Delta = 0$, then it has to be the case that u is matched to a vertex in some subtree T_i (not necessarily v_i), while any vertex from $N_{-i}(u) := \{v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_\Delta\}$ is matched with another vertex from $N_{-i}(u)$ (hence, we already get that Δ needs to be odd). This is because each subtree needs to have at least one vertex matched to a different subtree (since $|S_i|$ is odd for each $i \in [\Delta]$). If $M(u)$ is a vertex in T_i and, for any $v \in N_{-i}(u)$, we have $M(v) \in N_{-i}(u)$, we say that u orients the edge going to v_i outwards.

Consider a path $(u_0 = u, u_1, \ldots, u_k)$, where $k = 2t(n) + 3$. By the argument provided above, the probability that u orients an edge outwards is at least $1 - \Delta/n^2 - 1/n^{2\Delta}$, and, if u orients an edge outwards, the probability that this edge is not uu_1 is $(\Delta - 1)/\Delta$, by symmetry. Hence, we obtain (for sufficiently large n) that $\mathrm{P}(\mathcal{E}_1) \geq 1/2$, where $\mathcal{E}_1$ denotes the event that vertex u orients an edge different from uu_1 outwards. With an analogous argument, we obtain that $\mathrm{P}(\mathcal{E}_2) \geq 1/2$, where $\mathcal{E}_2$ denotes the event that vertex u_k orients an edge different from $u_k u_{k-1}$ outwards. Since $\mathcal{E}_1$ and $\mathcal{E}_2$ are independent (due to the fact that the distance between any neighbor of u and any neighbor of u_k is at least $2t(n)+1$), we obtain $\mathrm{P}(\mathcal{E}_1 \cap \mathcal{E}_2) \geq 1/4$. Moreover, the probability that all vertices in $\{u_1, \ldots, u_{k-1}\}$ orient an edge outwards is at least $1 - (2t(n)+2)(\Delta/n^2 + 1/n^{2\Delta})$, which is at least $4/5$ for sufficiently large n. Thus, by a union bound, there is a probability of at least $1 - 3/4 - 1/5 - 1/n^{2\Delta}$ that $\mathcal{A}$ does not fail, all vertices in $(u, u_1, \ldots, u_k)$ orient an edge outwards, and the outward oriented edges chosen by u and u_k are not uu_1 and $u_k u_{k-1}$, respectively. For sufficiently large n, the indicated probability is strictly larger than 0. We will obtain a contradiction and conclude the proof for Π^2_{pm} by showing that there is no correct solution with the described properties.

Assume in the following that the solution provided by $\mathcal{A}$ is correct and $u, u_1, \ldots, u_k$ satisfy the mentioned properties. Since u orients an edge different from uu_1 outwards, u_1 must be matched to a neighbor of u, which implies that u_1 orients edge $u_1 u$ outwards. Using an analogous argumentation, we obtain inductively that u_j orients edge $u_j u_{j-1}$ outwards, for all $2 \leq j \leq k$. However, this yields a contradiction to the fact that the edge that u_k orients outwards is different from $u_k u_{k-1}$, concluding the proof (for Π^2_{pm}).

The theorem statement for $\overline{\Pi^2_{\mathrm{pm}}}$ follows by applying Proposition 9.

□

5.4 Upper Bounds Using Forest Decompositions

We prove an upper bound for the problems Π^2_{pm} and $\overline{\Pi^2_{\mathrm{pm}}}$ defined in the previous subsection. We use a technique of decomposing the input graph in a spanning forest with some additional properties, i.e., one or two ended. This technique was used in [39] to prove Brooks' theorem in a measurable context. Namely, they proved that if $\mathcal{G}$ is a Borel Δ-regular acyclic graph and μ is a Borel probability measure, then it is possible to erase some edges of $\mathcal{G}$ in such a way that the remaining graph is a one-ended spanning forest on a μ-conull set. It is not hard to see that one can solve Π^2_{pm} on one-ended trees in an inductive manner, starting with the leaf vertices. Consequently we have $\Pi^2_{\mathrm{pm}} \in \mathsf{MEASURE}$. We provide a quantitative version of this result and thereby show that

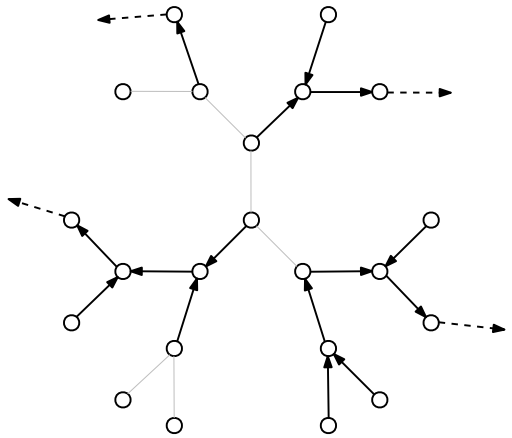

Figure 5: An example of a one-ended forest decomposition of an infinite 3-regular tree.

the problem of constructing a one-ended forest is contained in $\mathsf{OLOCAL}(\text{poly}\log 1/\varepsilon)$. A variation of the construction shows that it is possible to find a one or two-ended spanning forest in a Borel way. This allows us to show that $\overline{\Pi^2_{\text{pm}}} \in \mathsf{BOREL}$. As an application of the decomposition technique we show that Vizing's theorem, i.e., proper $\Delta+1$ edge coloring $\Pi_{\chi',\Delta+1}$, for $\Delta = 3$ is in the class $\mathsf{OLOCAL}(\text{poly}\log 1/\varepsilon)$.

5.4.1 Uniform Complexity of the One-Forest Decomposition

We start with the definition of a one-ended spanning forest. It can be viewed as a variation of the edge grabbing problem Π_{edgegrab}. Namely, if a vertex v grabs an incident edge e, then we think of e as being oriented away from v and v selecting the other endpoint of e as its parent. Suppose that $\mathcal{T}$ is a solution of Π_{edgegrab} on T_Δ. We denote with $\mathcal{T}^{\leftarrow}(v)$ the subtree with root v.

Definition 17 (One-ended spanning forest)**.** *We say that a solution $\mathcal{T}$ of $\Pi_{edgegrab}$ is a* one-ended spanning forest *if $\mathcal{T}^{\leftarrow}(v)$ is finite for every $v \in T_\Delta$ (see Figure 5).*

Note that on an infinite Δ-regular tree every connected component of a one-ended spanning forest must be infinite. Furthermore, recall that we discussed above that it is possible to construct a one-ended spanning forest in $\mathsf{MEASURE}$ by [39]. Next we formulate our quantitative version.

Theorem 12. *The one-ended spanning forest can be constructed by a uniform local algorithm with a uniform local complexity of $O(\text{poly}\log 1/\varepsilon)$. More precisely, there is a uniform distributed algorithm $\mathcal{A}$ that computes a one-ended spanning forest and if we define $R(v)$ to be the smallest coding radius that allows v to compute $\mathcal{T}^{\leftarrow}(v)$, then*

$$P(R(v) > O(\text{poly}\log 1/\varepsilon)) \leq \varepsilon.$$

The formal proof of Theorem 12 can be found in Section 5.5. We first show the main applications of the result. Namely, we show that Π^2_{pm} can be solved inductively starting from the leaves of the one-ended trees. This proves that Π^2_{pm} is in $\mathsf{OLOCAL}(\text{poly}\log 1/\varepsilon)$ and hence $\mathsf{OLOCAL}(\text{poly}\log 1/\varepsilon) \setminus \mathsf{RLOCAL}(o(\log n))$ is non-empty.

Theorem 13. *The problem Π^2_{pm} is in the class* $\mathsf{MEASURE}$ *and* $\mathsf{OLOCAL}(O(\text{poly}\log 1/\varepsilon))$.

Proof. First we show that every infinite one-ended directed tree T admits an inductive solution from the leaves to the direction of infinity. More precisely, we define a power-2 perfect matching on

the infinite one-ended directed tree T such that the following holds. Let v be an arbitrary vertex and $T^{\leftarrow}(v)$ the finite subtree rooted at v. Then, all vertices in $T^{\leftarrow}(v)$, with the possible exception of v, are matched to a vertex in $T^{\leftarrow}(v)$. Moreover, for each vertex in $T^{\leftarrow}(v) \setminus \{v\}$, it is possible to determine to which vertex it is matched by only considering the subtree $T^{\leftarrow}(v)$. We show that it is possible to define a perfect matching in that way by induction on the height of the subtree $T^{\leftarrow}(v)$.

We first start with the leaves and declare them as unmatched in the first step of the induction.

Now, consider an arbitrary vertex v with $R \geq 1$ children that we denote by $v_1, v_2, \ldots, v_R$. By the induction hypothesis, all vertices in $T^{\leftarrow}(v_i) \setminus \{v_i\}$ are matched with a vertex in $T^{\leftarrow}(v_i)$ for $i \in [R]$ and it is possible to compute that matching given $T^{\leftarrow}(v)$. However, some of the children of v are possibly still unmatched. If the number of unmatched children of v is even, then we simply pair them up in an arbitrary but fixed way and we leave v unmatched. Otherwise, if the number of unmatched children of v is odd, we pair up v with one of its unmatched children and we pair up all the remaining unmatched children of v in an arbitrary but fixed way. This construction clearly satisfies the criteria stated above. In particular, the resulting matching is a perfect matching, as every vertex gets matched eventually.

By Theorem 12 there is a uniform distributed algorithm of complexity $O(\operatorname{poly}\log 1/\varepsilon)$ that computes a one-ended spanning forest $\mathcal{T}$ on T_Δ. Moreover, every vertex $v \in T_\Delta$ can compute where it is matched with a uniform local complexity of $O(\operatorname{poly}\log 1/\varepsilon)$. This follows from the discussion above, as a vertex v can determine where it is matched once it has computed $\mathcal{T}^{\leftarrow}(w)$ for each w in its neighborhood. Hence, Π^2_{pm} is in the class $\mathsf{OLOCAL}(O(\operatorname{poly}\log 1/\varepsilon))$.

Similarly, the one-ended spanning forest $\mathcal{T} \subseteq \mathcal{G}$ can be constructed in the class $\mathsf{MEASURE}$ by [39]. By the discussion above, this immediately implies that Π^2_{pm} is in the class $\mathsf{MEASURE}$. □

5.4.2 Decomposition in BOREL

Next, we show that a similar, but weaker, decomposition can be done in a Borel way. Namely, we say that a subset of edges of an acyclic infinite graph G, denoted by $\mathcal{T}$, is a *one or two-ended spanning forest* if every vertex $v \in G$ is contained in an infinite connected component of $\mathcal{T}$ and each infinite connected component of $\mathcal{T}$ has exactly one or two directions to infinity. Let S be such a connected component. Note that if S has one end, then we can solve Π^2_{pm} on S as above. If S has two ends, then S contains a doubly infinite path P with the property that erasing P splits $S \setminus P$ into finite connected components. In order to solve $\overline{\Pi^2_{\mathrm{pm}}}$ we simply solve Π^2_{pm} on the finite connected components of $S \setminus P$ (with some of the vertices in $S \setminus P$ being potentially matched with vertices on the path P) and declare vertices on P to be line vertices.

The high-level idea to find a one or two-ended spanning forest is to do the same construction as in the measure case and understand what happens with edges that do not disappear after countably many steps. A slight modification in the construction guarantees that what remains are doubly infinite paths.

Theorem 14. *Let $\mathcal{G}$ be a Borel Δ-regular forest. Then there is a Borel one or two-ended spanning forest $\mathcal{T} \subseteq \mathcal{G}$.*

The proof of Theorem 14 can be found in Appendix C. We remark that the notation used in the proof is close to the notation in the proof that gives a measurable construction of a one-ended spanning forest in [39]. Next, we show more formally how Theorem 14 implies that $\overline{\Pi^2_{\mathrm{pm}}} \in \mathsf{BOREL}$.

Theorem 15. *The problem $\overline{\Pi^2_{pm}}$ is in the class* BOREL.

Proof. Let $\mathcal{T}$ be the one or two-ended spanning forest given by Theorem 14 and S be a connected component of $\mathcal{T}$. If S is one-ended, then we use the first part of the proof of Theorem 13 to find

a solution of Π^2_{pm} on S. Since deciding that S is one-ended as well as computing an orientation towards infinity can be done in a Borel way, this yields a Borel labeling.

Suppose that S is two-ended. Then, there exists a doubly infinite path P in S with the property that the connected components of $S \setminus P$ are finite. Moreover, it is possible to detect P in a Borel way. That is, declaring the vertices on P to be line vertices and using the special symbol in $\overline{\Pi^2_{\text{pm}}}$ for the half-edges on P yields a Borel measurable labeling. Let $C \neq \emptyset$ be one of the finite components in $S \setminus P$ and $v \in C$ be the vertex of distance 1 from P. Orient the edges in C towards v, this can be done in a Borel way since v is uniquely determined for C. Then the first part of the proof of Theorem 13 shows that one can inductively find a solution to Π^2_{pm} on C in such a way that all vertices, possibly up to v, are matched. If v is matched, then we are done. If v is not matched, then we add the edge that connects v with P to the matching. Note that it is possible that multiple vertices are matched with the same path vertex, but this is not a problem according to the definition of $\overline{\Pi^2_{\text{pm}}}$. It follows that this defines a Borel function on $H(\mathcal{G})$ that solves $\overline{\Pi^2_{\text{pm}}}$. □

5.4.3 Vizing's Theorem for $\Delta = 3$

Finding a measurable or local version of Vizing's Theorem, i.e., proper edge $(\Delta + 1)$-coloring $\Pi_{\chi',\Delta+1}$, was studied recently in [60, 107, 21]. It is however not known, even on trees, whether $\Pi_{\chi',\Delta+1}$ is in $\mathsf{RLOCAL}(O(\log \log n))$. Here we use the one-ended forest construction to show that $\Pi_{\chi',\Delta+1}$ is in the class $\mathsf{OLOCAL}(\text{poly} \log 1/\varepsilon)$ for $\Delta = 3$.

Proposition 11. *Let $\Delta = 3$. We have $\Pi_{\chi',\Delta+1} \in \mathsf{OLOCAL}(\text{poly} \log 1/\varepsilon)$.*

Proof sketch. By Theorem 12, we can compute a one-ended forest decomposition $\mathcal{T}$ with uniform local complexity $O(\text{poly} \log 1/\varepsilon)$. Note that every vertex has at least one edge in $\mathcal{T}$, hence the edges in $G \setminus \mathcal{T}$ form paths. These paths can be 3-edge colored by using the uniform version of Linial's coloring algorithm. This algorithm has a uniform local complexity of $O(\log^* 1/\varepsilon)$. By Lemma 1, the overall complexity is $O(\text{poly} \log(\Delta^{\log^* 1/\varepsilon}/\varepsilon) + \log^* 1/\varepsilon)) = O(\text{poly} \log 1/\varepsilon)$.

Finally, we color the edges of T. We start from the leaves and color the edges inductively. In particular, whenever we consider a vertex v we color the at most two edges in $T^{\leftarrow}(v)$ below v. Note that there always is at least one uncolored edge going from v in the direction of T. Hence, we can color the at most two edges below v in $T^{\leftarrow}(v)$ greedily – each one neighbors with at most 3 colored edges at any time. □

5.5 Proof of Theorem 12

In this section we formally prove Theorem 12.

Theorem 12. *The one-ended spanning forest can be constructed by a uniform local algorithm with a uniform local complexity of $O(\text{poly} \log 1/\varepsilon)$. More precisely, there is a uniform distributed algorithm $\mathcal{A}$ that computes a one-ended spanning forest and if we define $R(v)$ to be the smallest coding radius that allows v to compute $\mathcal{T}^{\leftarrow}(v)$, then*

$$P(R(v) > O(\text{poly} \log 1/\varepsilon)) \leq \varepsilon.$$

We follow the construction of the one-ended spanning forest in [39]. The construction proceeds in rounds. Before each round, some subset of the vertices have already decided which incident edge to grab, and we refer to these vertices as settled vertices. All the remaining vertices are called unsettled. The goal in each round is to make a large fraction of the unsettled vertices settled. To

achieve this goal, our construction relies on certain properties that the graph induced by all the unsettled vertices satisfies.

One important such property is that the graph induced by all the unsettled vertices is expanding. That is, the number of vertices contained in the neighborhood around a given vertex grows exponentially with the radius. The intuitive reason why this is a desirable property is the following. Our algorithm will select a subset of the unsettled vertices and clusters each unsettled vertex to the closest selected vertex. As the graph is expanding and any two vertices in the selected subset are sufficiently far away, there will be a lot of edges leaving a given cluster. For most of these inter-cluster edges, both of the endpoints will become settled. This in turn allows one to give a lower bound on the fraction of vertices that become settled in each cluster.

To ensure that the graph induced by all the unsettled vertices is expanding, a first condition we impose on the graph induced by all the unsettled vertices is that it has a minimum degree of at least 2. While this condition is a first step in the right direction, it does not completely suffice to ensure the desired expansion, as an infinite path has a minimum degree of 2 but does not expand sufficiently. Hence, our algorithm will keep track of a special subset of the unsettled vertices, the so-called hub vertices. Each hub vertex has a degree of Δ in the graph induced by all the unsettled vertices. That is, all of the neighbors of a hub vertex are unsettled as well. Moreover, each unsettled vertex has a bounded distance to the closest hub vertex, where the specific upper bound on the distance to the closest hub vertex increases with each round. As we assume $\Delta > 2$, the conditions stated above suffice to show that the graph induced by all the unsettled vertices expands.

Next, we explain in more detail how a vertex decides which edge to grab. Concretely, if a vertex becomes settled, it grabs an incident edge such that the other endpoint of that edge is strictly closer to the closest unsettled vertex as the vertex itself. For example, if a vertex becomes settled and there is exactly one neighbor that is still unsettled, then the vertex will grab the edge that the vertex shares with its unsettled neighbor. Grabbing edges in that way, one can show two things. First, for a settled vertex v, the set $T^{\leftarrow}(v)$ of vertices behind v does not change once v becomes settled. The intuitive reason for this is that the directed edges point towards the unsettled vertices and therefore the unsettled vertices lie before v and not after v. Moreover, one can also show that $T^{\leftarrow}(v)$ only contains finitely many vertices. The reason for this is that at the moment a vertex v becomes settled, there exists an unsettled vertex that is sufficiently close to v, where the exact upper bound on that distance again depends on the specific round in which v becomes settled.

What remains to be discussed is at which moment a vertex decides to become settled. As written above, in each round the algorithm considers a subset of the unsettled vertices and each unsettled vertex is clustered to the closest vertex in that subset. This subset only contains hub vertices and it corresponds to an MIS on the graph with the vertex set being equal to the set of hub vertices and where two hub vertices are connected by an edge if they are sufficiently close in the graph induced by all the unsettled vertices. Now, each cluster center decides to connect to exactly Δ different neighboring clusters, one in each of its Δ subtrees. Remember that as each cluster center is a hub vertex, all of its neighbors are unsettled as well. Now, each unsettled vertex becomes settled except for those that lie on the unique path between two cluster centers such that one of the cluster centers decided to connect to the other one.

Construction The algorithm proceeds in rounds. After each round, a vertex is either settled or unsettled and a settled vertex remains settled in subsequent rounds. Moreover, some unsettled vertices are so-called hub vertices. We denote the set of unsettled, settled and hub vertices after the i-th round with U_i, S_i and H_i, respectively. We set $U_0 = H_0 = V$ prior to the first round and we always set $S_i = V \setminus U_i$. Figure 6 illustrates the situation after round i. Moreover, we denote

with O_i a partial orientation of the edges of the infinite Δ-regular input tree T. In the beginning, O_0 corresponds to the partial orientation with no oriented edges. Each vertex is incident to at most 1 outwards oriented edge in O_i. If a vertex is incident to an outwards oriented edge in O_i, then the other endpoint of the edge will be its parent in the one-ended forest decomposition. For each vertex v in S_i, we denote with $T_{v,i}$ the smallest set that satisfies the following conditions. First, $v \in T_{v,i}$. Moreover, if $u \in T_{v,i}$ and $\{w, u\}$ is an edge that according to O_i is oriented from w to u, then $w \in T_{v,i}$. We later show that $T_{v,i}$ contains exactly those vertices that are contained in the subtree $\mathcal{T}^{\leftarrow}(v)$.

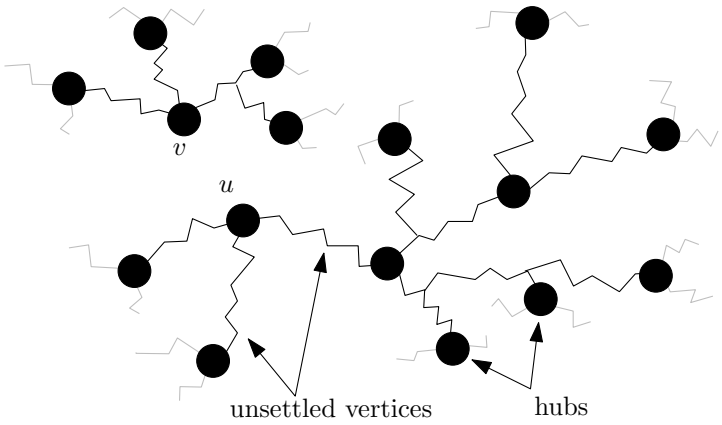

Figure 6: The picture illustrates the situation after some round i of the construction. The vertices on the paths are all contained in U_i, while the vertices corresponding to the big black dots are additionally contained in H_i. Note that the length of each path is at least d_i. However, the distance between u and v in the original graph can be much smaller.

After the i-th round, the construction satisfies the following invariants.

1. For each $v \in S_{i-1}$, we have $T_{v,i} = T_{v,i-1}$.
2. Let v be an arbitrary vertex in S_i. Then, $T_{v,i}$ contains finitely many vertices and furthermore $T_{v,i} \subseteq S_i$.
3. The minimum degree of the graph $T[U_i]$ is at least 2 and each vertex in H_i has a degree of Δ in $T[U_i]$.
4. Each vertex in U_i has a distance of at most $\sum_{j=0}^{i} d_j$ to the closest vertex in H_i in the graph $T[U_i]$.

We now describe how to compute U_i, S_i, H_i and O_i. The construction during the i-th round is illustrated in *Figure* 7. Note that we can assume that the invariants stated above are satisfied after the $(i-1)$-th round. In the i-th round we have a parameter d_i that we later set to 2^{2^i}.

1. H_i is a subset of H_{i-1} that satisfies the following property. No two vertices in H_i have a distance of at most d_i in $T[U_{i-1}]$. Moreover, each vertex in $H_{i-1} \setminus H_i$ has a distance of at most d_i to the closest vertex in H_i in the graph $T[U_{i-1}]$. Note that we can compute H_i by computing an MIS in the graph with vertex set H_{i-1} and where two vertices are connected iff they have a distance of at most d_i in the graph $T[U_{i-1}]$. Hence, we can use Ghaffari's MIS algorithm [57] to compute the set H_i.

2. Next, we describe how to compute U_i. We assign each vertex $u \in U_{i-1}$ to the closest vertex in the graph $T[U_{i-1}]$ that is contained in H_i, with ties being broken arbitrarily. We note that there exists a node in H_i with a distance of at most $(\sum_{j=0}^{i-1} d_j) + d_i$ to u. To see why, note that Invariant (4) from round $i-1$ implies that there exists a vertex w in H_{i-1} with a distance of at most $\sum_{j=0}^{i-1} d_j$ to u. We are done if w is also contained in H_i. If not, then it follows from the way we compute H_i that there exists a vertex in H_i with a distance of at most d_i to w, but then the triangle inequality implies that the distance from u to that vertex is at most $(\sum_{j=0}^{i-1} d_j) + d_i$, as desired. For each vertex $v \in H_i$, we denote with $C(v)$ the set of all vertices in U_{i-1} that got assigned to v. Now, let E_v denote the set of edges that have exactly one endpoint in $C(v)$. We can partition E_v into $E_{v,1} \sqcup E_{v,2} \sqcup \ldots \sqcup E_{v,\Delta}$, where for $\ell \in [\Delta]$, $E_{v,\ell}$ contains all the edges in E_v that are contained in the ℓ-th subtree of v. Invariants (3) and (4) after the $(i-1)$-th round imply that $E_{v,\ell} \neq \emptyset$. Moreover, $E_{v,\ell}$ contains only finitely many edges, as we have shown above that the cluster radius is upper bounded by $\sum_{j=0}^{i} d_j$.

   Now, for $\ell \in [\Delta]$, we choose an edge e_ℓ uniformly at random from the set $E_{v,\ell}$. Let $u_\ell \neq v$ be the unique vertex in H_i such that one endpoint of e_ℓ is contained in $C(u_\ell)$. We denote with $P_{v,\ell}$ the set of vertices that are contained in the unique path between v and u_ℓ. Finally, we set $U_i = \bigcup_{v \in H_i, \ell \in [\Delta]} P_{v,\ell}$.

3. It remains to describe how to compute the partial orientation O_i. All edges that are oriented in O_{i-1} will be oriented in the same direction in O_i. Additionally, we orient for each vertex u that got settled in the i-th round, i.e., $u \in S_i \cap U_{i-1}$, exactly one incident edge away from u. Let w be the closest vertex to u in $T[U_{i-1}]$ that is contained in U_i, with ties being broken arbitrarily. We note that it follows from the discussions above that the distance between u and w in the graph $T[U_{i-1}]$ is at most $\sum_{j=0}^{i} d_j$. We now orient the edge incident to u that is on the unique path between u and w outwards. We note that this orientation is well-defined in the sense that we only orient edges that were not oriented before and that we don't have an edge such that both endpoints of that edge want to orient the edge outwards.

We now prove by induction that our procedure satisfies all the invariants. Prior to the first round, all invariants are trivially satisfied. Hence, it remains to show that the invariants hold after the i-th round, given that the invariants hold after the $(i-1)$-th round.

1. Let $v \in S_{i-1}$ be arbitrary. As O_i is an extension of O_{i-1}, it follows from the definition of $T_{v,i-1}$ and $T_{v,i}$ that $T_{v,i-1} \subseteq T_{v,i}$. Now assume that $T_{v,i} \not\subseteq T_{v,i-1}$. This would imply the existence of an edge $\{u, w\}$ such that $u \in T_{v,i-1}$ and the edge was oriented during the i-th round from w to u. As $u \in T_{v,i-1}$, Invariant (2) from the $(i-1)$-th round implies $u \in S_{i-1}$. This is a contradiction as during the i-th round only edges with both endpoints in U_{i-1} can be oriented. Therefore it holds that $T_{v,i} = T_{v,i-1}$, as desired.

2. Let $v \in S_i$ be arbitrary. If it also holds that $v \in S_{i-1}$, then the invariant from round $i-1$ implies that $T_{v,i-1} = T_{v,i}$ contains finitely many vertices and $T_{v,i-1} \subseteq S_{i-1} \subseteq S_i$. Thus, it suffices to consider the case that $v \in S_i \cap U_{i-1}$. Note that $T_{v,i} \cap U_i = \emptyset$. Otherwise there would exist an edge that is oriented away from a vertex in U_i according to O_i, but this cannot happen according to the algorithm description. Hence, $T_{v,i} \subseteq S_i$. Now, for the sake of contradiction, assume that $T_{v,i}$ contains infinitely many vertices. From the definition of $T_{v,i}$, this implies that there exists a sequence of vertices $(v_k)_{k \geq 1}$ with $v = v_1$ such that for each $k \geq 1$, $\{v_{k+1}, v_k\}$ is an edge in T that is oriented from v_{k+1} to v_k according to O_i. For each $k \geq 1$ we have $v_k \in S_i$. We furthermore know that $v_k \in U_{i-1}$, as otherwise $T_{v_k,i} = T_{v_k,i-1}$ would contain infinitely

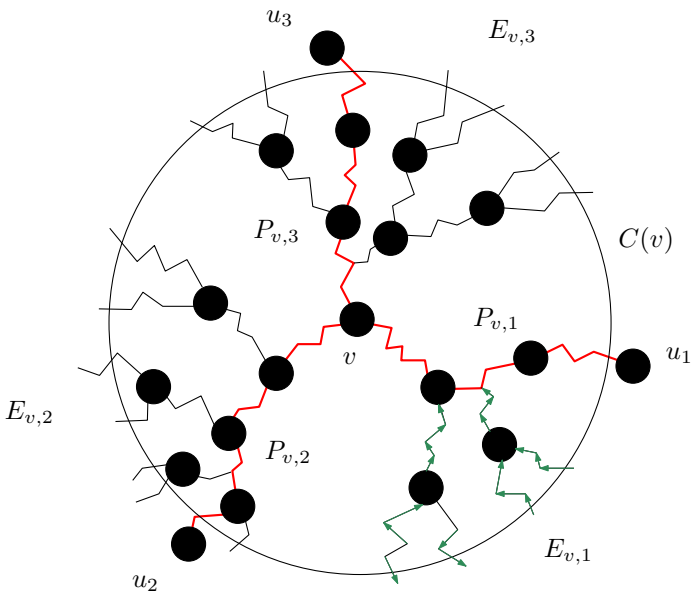

Figure 7: The picture illustrates the situation during some round i of the construction. The vertices on all paths are contained in U_{i-1} and the vertices corresponding to the big black dots are contained in H_{i-1}. The vertices v, u_1, u_2 and u_3 are the only vertices in the picture that are contained in H_i. All the vertices on the red paths remain unsettled during round i while all the vertices on the black paths become settled. Each vertex that becomes settled during the i-th round orients one incident edge outwards, illustrated by the green edges. Each green edge is oriented towards the closest vertex that is still unsettled after the i-th round. We note that some of the edges in the picture are oriented away from v.

many vertices, a contradiction. From the way we orient the edges, a simple induction proof implies that the closest vertex of v_k in the graph $T[U_{i-1}]$ that is contained in U_i has a distance of at least k. However, we previously discussed that the distance between v_k and the closest vertex in U_i in the graph $T[U_{i-1}]$ is at most $\sum_{j=0}^{i} d_j$. This is a contradiction. Hence, $T_{v,i}$ contains finitely many vertices.

3. It follows directly from the description of how to compute U_i and H_i that the minimum degree of the graph $T[U_i]$ is at least 2 and that each vertex in H_i has a degree of Δ in $T[U_i]$.

4. Let $u \in U_i$ be arbitrary. We need to show that u has a distance of at most $\sum_{j=0}^{i} d_j$ to the closest vertex in H_i in the graph $T[U_i]$. Let v be the vertex with $u \in C(v)$. We know that there is no vertex in H_i that is closer to u in $T[U_{i-1}]$ than v. Hence, the distance between u and v is at most $\sum_{j=0}^{i} d_j$ in the graph $T[U_{i-1}]$. Moreover, from the description of how we compute U_i, it follows that all the vertices on the unique path between u and v are contained in U_i. Hence, each vertex in U_i has a distance of at most $\sum_{j=0}^{i} d_j$ to the closest vertex in H_i in the graph $T[U_i]$, as desired.

We derive an upper bound on the coding radius of the algorithm in three steps. First, for each $i \in \mathbb{N}$ and $\varepsilon > 0$, we derive an upper bound on the coding radius for computing with probability at least $1 - \varepsilon$ for a given vertex in which of the sets S_i, U_i and H_i it is contained in and for each incident edge whether and how it is oriented in O_i, given that each vertex receives as additional input in which of the sets S_{i-1}, U_{i-1} and H_{i-1} it is contained in and for each incident edge whether and how it is oriented in O_{i-1}. Given this upper bound on the coding radius, we can use the sequential composition lemma (Lemma 1) and a simple induction proof to give an upper bound on the coding radius for computing with probability at least $1 - \varepsilon$ for a given vertex in which of the sets S_i, U_i and H_i it is contained in and for each incident edge whether and how it is oriented in O_i, this time without providing any additional input.

Second, we analyze after how many rounds a given vertex is settled with probability at least $1 - \varepsilon$ for a given $\varepsilon > 0$.

Finally, we combine these two upper bounds to prove Theorem 12.

Lemma 3. *For each $i \in \mathbb{N}$ and $\varepsilon \in (0, 0.01]$, let $g_i(\varepsilon)$ denote the smallest coding radius such that with probability at least $1 - \varepsilon$ one knows for a given vertex in which of the sets S_i, U_i and H_i it is contained in and for each incident edge whether and how it is oriented in O_i, given that each vertex receives as additional input in which of the sets S_{i-1}, U_{i-1} and H_{i-1} it is contained in and for each incident edge whether and how it is oriented in O_{i-1}. It holds that $g_i(\varepsilon) = O(d_i^2 \log(\Delta/\varepsilon))$.*

Proof. When running Ghaffari's uniform MIS algorithm on some graph G' with maximum degree Δ', each vertex knows with probability at least $1 - \varepsilon$ whether it is contained in the MIS or not after $O(\log(\Delta') + \log(1/\varepsilon))$ communication rounds in G' ([57], Theorem 1.1). For the MIS computed during the i-th round, $\Delta' = \Delta^{O(d_i)}$ and each communication round in G' can be simulated with $O(d_i)$ communication rounds in the tree T. Hence, to know for a given vertex with probability at least $1-\varepsilon$ whether it is contained in the MIS or not, it suffices consider the $O(d_i^2 \log(\Delta) + d_i \log(1/\varepsilon))$-hop neighborhood around that vertex. Now, let u be an arbitrary vertex. In order to compute in which of the sets S_i, U_i and H_i u is contained in and for each incident edge of u whether and how it is oriented in O_i, we not only need to know whether u is in the MIS or not. However, it suffices if we know for all the vertices in the $O(d_i)$-hop neighborhood of u whether they are contained in the MIS or not (on top of knowing for each vertex in the neighborhood in which of the sets S_{i-1}, U_{i-1} and H_{i-1} it is contained in and for each incident edge whether and how it is oriented in O_{i-1}).

Hence, by a simple union bound over the at most $\Delta^{O(d_i)}$ vertices in the $O(d_i)$-hop neighborhood around u, we obtain $g_i(\varepsilon) = O(d_i) + O(d_i^2 \log(\Delta) + d_i \log(\Delta^{O(d_i)}/\varepsilon)) = O(d_i^2 \log(\Delta/\varepsilon))$. □

Lemma 4. *For each $i \in \mathbb{N}$ and $\varepsilon \in (0, 0.01]$, let $h_i(\varepsilon)$ denote the smallest coding radius such that with probability at least $1-\varepsilon$ one knows for a given vertex in which of the sets S_i, U_i and H_i it is contained in and for each incident edge whether and how it is oriented in O_i. Then, there exists a constant c independent of Δ such that $h_i(\varepsilon) \leq 2^{2^{i+2}} \cdot (c \log(\Delta))^i \log(\Delta/\varepsilon)$.*

Proof. By prove the statement by induction on i. For a large enough constant c, it holds that $h_1(\varepsilon) \leq 2^{2^3} \cdot (c \log(\Delta)) \log(\Delta/\varepsilon)$. Now, consider some arbitrary i and assume that $h_i(\varepsilon) \leq 2^{2^{i+2}} \cdot (c \log(\Delta))^i \log(\Delta/\varepsilon)$ for some large enough constant c. We show that this implies $h_{i+1}(\varepsilon) \leq 2^{2^{(i+1)+2}} \cdot (c \log(\Delta))^{i+1} \log(\Delta/\varepsilon)$. By the sequential composition lemma (Lemma 1) and assuming that c is large enough, we have

$$\begin{aligned}
h_{i+1}(\varepsilon) &\leq h_i((\varepsilon/2)/\Delta^{g_{i+1}(\varepsilon/2)+1}) + g_{i+1}(\varepsilon/2) \\
&\leq 2^{2^{i+2}} \cdot (c \log(\Delta))^i \log(\Delta \cdot \Delta^{g_{i+1}(\varepsilon/2)+1}/(\varepsilon/2)) + g_{i+1}(\varepsilon/2) \\
&\leq 2 \cdot 2^{2^{i+2}} \cdot (c \log(\Delta))^i \log(\Delta \cdot \Delta^{g_{i+1}(\varepsilon/2)+1}/(\varepsilon/2)) \\
&\leq 2 \cdot 2^{2^{i+2}} \cdot (c \log(\Delta))^i \log(\Delta^{(c/10) \cdot 2^{2^{i+2}} \log(\Delta/\varepsilon)}/(\varepsilon/2)) \\
&\leq 4 \cdot 2^{2^{i+2}} \cdot (c \log(\Delta))^i \log(\Delta^{(c/10) \cdot 2^{2^{i+2}} \log(\Delta/\varepsilon)}) \\
&\leq (4/10) \cdot 2^{2^{i+2}} \cdot 2^{2^{i+2}} (c \log(\Delta))^{i+1} \log(\Delta/\varepsilon) \\
&\leq 2^{2^{(i+1)+2}} \cdot (c \log(\Delta))^{i+1} \log(\Delta/\varepsilon),
\end{aligned}$$

as desired.

□

Lemma 5. *Let u be an arbitrary vertex. For each $\varepsilon \in (0, 0.01]$, let $f(\varepsilon)$ denote the smallest $i \in \mathbb{N}$ such that u is settled after the i-th round with probability at least $1-\varepsilon$. There exists a fixed $c \in \mathbb{R}$ independent of Δ such that $f(\varepsilon) \leq \lceil 1 + \log\log \frac{1}{c} \log_\Delta(1/\varepsilon) \rceil$.*

Proof. Let $i \in \mathbb{N}$ be arbitrary. We show that a given vertex is settled after the i-th round with probability at least $1 - O(1/\Delta^{\Omega(d_i/d_{i-1})})$. For the sake of analysis, we run the algorithm on a finite Δ-regular high-girth graph instead of an infinite Δ-regular tree. For now, we additionally assume that no vertex realizes that we don't run the algorithm on an infinite Δ-regular tree. That is, we assume that for each vertex the coding radius to compute all its local information after the i-th round is much smaller than the girth of the graph.

With this assumption, we give a deterministic upper bound on the fraction of vertices that are not settled after the i-th round. On the one hand, the number of vertices that are not settled after the i-th round can be upper bounded by $|H_i| \cdot \Delta \cdot O(d_i)$. On the other hand, we will show that the fraction of vertices that are contained in H_i is upper bounded by $1/\Delta^{\Omega(d_i/d_{i-1})}$. We do this by showing that $C(v)$ contains $\Delta^{\Omega(d_i/d_{i-1})}$ many vertices for a given $v \in H_i$. Combining these two bounds directly implies that the fraction of unsettled vertices is smaller than

$$\frac{\Delta \cdot O(d_i)}{\Delta^{\Omega(d_i/d_{i-1})}} = 1/\Delta^{\Omega(d_i/d_{i-1})}.$$

Let $v \in H_i$ be arbitrary. We show that $|C(v)| = \Delta^{\Omega(d_i/d_{i-1})}$. Using Invariants (3) and (4) together with a simple induction argument, one can show that there are at least $(\Delta-1)^{\lfloor D/((2\sum_{j=0}^{i-1} d_j)+1)\rfloor}$

vertices contained in H_{i-1} and whose distance to v is at most D in the graph induced by the vertices in U_{i-1}. Furthermore, from the way we defined the clustering $C(v)$ and the fact that two vertices in H_i have a distance of at least d_i in the graph $T[U_{i-1}]$, it follows that all vertices in U_{i-1} having a distance of at most $d_i/2-1$ to v are contained in the cluster $C(v)$. Hence, the total number of vertices in $C(v)$ is at least $(\Delta-1)^{\lfloor (d_i/2-1)/((2\sum_{j=0}^{i-1} d_j)+1)\rfloor} = \Delta^{\Omega(d_i/d_{i-1})}$, as promised.

Now we remove the assumption that no vertex realizes that we don't run the algorithm on an infinite Δ-regular tree. If a vertex does realize that we don't run the algorithm on an infinite Δ-regular tree, then we consider the vertex as being unsettled after the i-th round. By considering graphs with increasing girth, we can make the expected fraction of vertices that realize that we are not on an infinite Δ-regular tree arbitrarily small. Combining this observation with the previous discussion, this implies that the expected fraction of vertices that are not settled after the i-th round is at most $1/\Delta^{\Omega(d_i/d_{i-1})}$. By symmetry, each vertex has the same probability of being settled after the i-th round. Hence, the probability that a given vertex is settled after the i-th round when run on a graph with sufficiently large girth is $1-1/\Delta^{\Omega(d_i/d_{i-1})}$, and the same holds for each vertex when we run the algorithm on an infinite Δ-regular tree. Thus, there exists a constant c such that the probability that a given vertex is unsettled after the i-th round is at most $1/\Delta^{c\cdot d_i/d_{i-1}} = 1/\Delta^{c\cdot 2^{2^{i-1}}}$. Setting $i = \lceil 1+\log\log\frac{1}{c}\log_\Delta(1/\varepsilon)\rceil$ finishes the proof. □

We are now finally ready to finish the proof of Theorem 12. Let u be an arbitrary vertex and $\varepsilon \in (0, 0.01]$. We need to compute an upper bound on the coding radius that is necessary for u to know all the vertices in $\mathcal{T}^{\leftarrow}(u)$ with probability at least $1-\varepsilon$. To compute such an upper bound for the required coding radius it suffices to find an i^* and an R^* such that the following holds. First, u is settled after i^* rounds with probability at least $1-\varepsilon/2$. Second, u knows for each edge in its $O(d_{i^*})$-hop neighborhood whether it is oriented in the partial orientation O_{i^*}, and if yes, in which direction, by only considering its R^*-hop neighborhood with probability at least $1-\varepsilon/2$. By a union bound, both of these events occur with probability at least $1-\varepsilon$. Moreover, if both events occur u knows all the vertices in the set $\mathcal{T}^{\leftarrow}(u)$. The reason is as follows. If u is settled after round i^*, then it follows from the previous analysis that only vertices in the $O(d_{i^*})$-hop neighborhood of u can be contained in $\mathcal{T}^{\leftarrow}(u)$. Moreover, for each vertex in its $O(d_{i^*})$-hop neighborhood, u can determine if the vertex is contained in $\mathcal{T}^{\leftarrow}(u)$ if it knows all the edge orientations on the unique path between itself and that vertex after the i^*-th round. Hence, it remains to find concrete values for i^* and R^*. According to Lemma 5, we can choose $i^* = \lceil 1+\log\log\frac{1}{c}\log_\Delta(2/\varepsilon)\rceil$ for some large enough constant c. Moreover, it follows from a union bound that all vertices in the $O(d_{i^*})$-hop neighborhood around u know with probability at least $1-\varepsilon/2$ the orientation of all its incident edges according to O_{i^*} by only considering their $h_{i^*}((\varepsilon/2)/\Delta^{O(d_{i^*})})$-hop neighborhood. Hence, we can set

$$\begin{aligned}
R^* &= O(d_{i^*}) + h_{i^*}((\varepsilon/2)/\Delta^{O(d_{i^*})}) \\
&\leq O(d_{i^*}) + 2^{2^{i^*+2}} \cdot (c\log(\Delta))^{i^*}\log(\Delta^{O(d_{i^*})}/\varepsilon) \\
&= O(2^{2^{i^*}} \cdot 2^{2^{i^*+2}}(c\log(\Delta))^{i^*}\log(\Delta/\varepsilon)) \\
&= O(2^{2^{i^*+3}}(c\log(\Delta))^{i^*}\log(\Delta/\varepsilon)) \\
&= O(2^{2^{\log\log\frac{1}{c}\log(2/\varepsilon)+5}}(c\log(\Delta))^{\log\log\frac{1}{c}\log(2/\varepsilon)+2}\log(\Delta/\varepsilon)) \\
&= \log(1/\varepsilon)^{32}\cdot\log(\Delta/\varepsilon)\cdot\log\log(1/\varepsilon)^{O(\log\log(\Delta))}.
\end{aligned}$$

As $\Delta = O(1)$, it therefore holds that

$$P(R(v)) = \text{poly}(\log(1/\varepsilon)) \geq 1 - \varepsilon,$$

as desired.

6 BAIRE = LOCAL($O(\log n)$)

In this section, we show that on Δ-regular trees the classes BAIRE and LOCAL($O(\log(n))$) are the same. At first glance, this result looks rather counter-intuitive. This is because in finite Δ-regular trees every vertex can see a leaf of distance $O(\log(n))$, while there are no leaves at all in an infinite Δ-regular tree. However, there is an intuitive reasons why these classes are the same: in both setups there is a technique to decompose an input graph into a hierarchy of subsets. Furthermore, the existence of a solution that is defined inductively with respect to these decompositions can be characterized by the same combinatorial condition of Bernshteyn [19]. We start with a high-level overview of the decomposition techniques used in both contexts.

Rake and Compress The hierarchical decomposition in the context of distributed computing is based on a variant of a decomposition algorithm of Miller and Reif [92]. Their original decomposition algorithm works as follows. Start with a tree T, and repeatedly apply the following two operations alternately: Rake (remove all degree-1 vertices) and Compress (remove all degree-2 vertices). Then $O(\log n)$ iterations suffice to remove all vertices in T [92]. To view it another way, this produces a decomposition of the vertex set V into $2L-1$ layers

$$V = V_1^{\mathsf{R}} \cup V_1^{\mathsf{C}} \cup V_2^{\mathsf{R}} \cup V_2^{\mathsf{C}} \cup V_3^{\mathsf{R}} \cup V_3^{\mathsf{C}} \cup \cdots \cup V_L^{\mathsf{R}},$$

with $L = O(\log n)$, where V_i^{R} is the set of vertices removed during the i-th Rake operation and V_i^{C} is the set of vertices removed during the i-th Compress operation. We will use a variant [35] of this decomposition in the proof of Proposition 14.

Variants of this decomposition turned out to be useful in designing LOCAL algorithms [35, 33, 32]. In our context, we assume that the given LCL satisfies a certain combinatorial condition and then find a solution inductively, in the reversed order of the construction of the decomposition. Namely, in the Rake step we want to be able to existentially extend the inductive partial solution to all relative degree 1-vertices (each $v \in V_i^{\mathsf{R}}$ has degree at most 1 in the subgraph induced by $V_i^{\mathsf{R}} \cup \cdots \cup V_L^{\mathsf{R}}$) and in the Compress step we want to extend the inductive partial solution to paths with endpoints labeled from the induction (the vertices in V_i^{C} form degree-2 paths in the subgraph induced by $V_i^{\mathsf{C}} \cup \cdots \cup V_L^{\mathsf{R}}$).

TOAST Finding a hierarchical decomposition in the context of descriptive combinatorics is tightly connected with the notion of *Borel hyperfiniteness*. Understanding what Borel graphs are Borel hyperfinite is a major theme in descriptive set theory [46, 54, 38]. It is known that grids, and generally polynomial growth graphs are hyperfinite, while, e.g., acyclic graphs are not in general hyperfinite [74]. A strengthening of hyperfiniteness that is of interest to us is called a *toast* [55, 41]. A q-toast, where $q \in \mathbb{N}$, of a graph G is a collection $\mathcal{D}$ of finite subsets of G with the property that (i) every pair of vertices is covered by an element of $\mathcal{D}$ and (ii) the boundaries of every $D \neq E \in \mathcal{D}$ are at least q apart. The idea to use a toast structure to solve LCLs appears in [41] and has many applications since then [55, 87]. This approach has been formalized in [62], where the authors introduce TOAST algorithms. Roughly speaking, an LCL Π admits a TOAST algorithm if there is $q \in \mathbb{N}$ and a partial extending function (the function is given a finite subset of a tree that is

partially colored and outputs an extension of this coloring on the whole finite subset) that has the property that whenever it is applied inductively to a q-toast, then it produces a Π-coloring. An advantage of this approach is that once we know that a given Borel graph admits, e.g., a Borel toast structure and a given LCL Π admits a TOAST algorithm, then we may conclude that Π is in the class BOREL. Similarly for MEASURE, BAIRE or ULOCAL, we refer the reader to [62] for more details and results concerning grids.

In the case of trees there is no way of constructing a Borel toast in general, however, it is a result of Hjorth and Kechris [66] that every Borel graph is hyperfinite on a comeager set for every compatible Polish topology. A direct consequence of [84, Lemma 3.1] together with a standard construction of toast via Voronoi cells gives the following strengthening to toast. We include a sketch of the proof for completeness.

Proposition 12. *Let $\mathcal{G}$ be a Borel graph on a Polish space (X, τ) with degree bounded by $\Delta \in \mathbb{N}$. Then for every $q > 0$ there is a Borel $\mathcal{G}$-invariant τ-comeager set C on which $\mathcal{G}$ admits a Borel q-toast.*

Proof sketch. Let $\{A_n\}_{n\in\mathbb{N}}$ be a sequence of MIS with parameter $f(n)$ as in [84, Lemma 3.1] for a sufficiently fast growing function $f(n)$, e.g., $f(n) = (2q)^{n^2}$. Then $C = \bigcup_{n\in\mathbb{N}} A_n$ is a Borel τ-comeager set that is $\mathcal{G}$-invariant. We produce a toast structure in a standard way, e.g., see [62, Appendix A].

Let $B_n(x)$ denote the ball of radius $f(n)/3$ and $R_n(x)$ the ball of radius $f(n)/4$ around $x \in A_n$. Iteratively, define cells $\mathcal{D}_n = \{C_n(x)\}_{x\in A_n}$ as follows. Set $C_1(x) = R_1(x)$. Suppose that $\mathcal{D}_n$ has been defined and set

- $H^{n+1}(x, n+1) := R_{n+1}(x)$ for every $x \in A_{n+1}$,
- if $1 \le i \le n$ and $\left\{H^{n+1}(x, i+1)\right\}_{x\in A_{k+1}}$ has been defined, then we put

$$H^{n+1}(x,i) = \bigcup \left\{B_i(y) : H^{n+1}(x, i+1) \cap B_i(y) \neq \emptyset\right\}$$

  for every $x \in A_{n+1}$,
- set $C_{n+1}(x) := H^{n+1}(x, 1)$ for every $x \in A_{n+1}$, this defines $\mathcal{D}_{n+1}$.

The fact that $\mathcal{D} = \bigcup_{n\in\mathbb{N}} \mathcal{D}_n$ is a q-toast on C follows from the fact that $C = \bigcup_{n\in\mathbb{N}} A_n$ together with the fact that the boundaries are q separated which can be shown as in [62, Appendix A]. □

Therefore to understand LCLs in the class BAIRE we need understand what LCLs on trees admit TOAST algorithm. It turns out that these notions are equivalent, again by using the combinatorial characterization of Bernshteyn [19] that we now discuss.

Combinatorial Condition – ℓ-full set In both decompositions, described above, we need to extend a partial coloring along paths that have their endpoints colored from the inductive step. The precise formulation of the combinatorial condition that captures this demand was extracted by Bernshteyn [19]. He proved that it characterizes the class BAIRE for Cayley graphs of virtually free groups. Note that this class contains, e.g., Δ-regular trees with a proper edge Δ-coloring.

Definition 18 (Combinatorial condition – an ℓ-full set)**.** *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL and $\ell \ge 2$. A set $\mathcal{V}' \subseteq \mathcal{V}$ is* ℓ-full *whenever the following is satisfied. Take a path with at least ℓ vertices, and add half-edges to it so that each vertex has degree Δ. Take any $c_1, c_2 \in \mathcal{V}'$ and label arbitrarily the half-edges around the endpoints with c_1 and c_2, respectively. Then there is a way to label the half-edges around the remaining $\ell - 2$ vertices with configurations from $\mathcal{V}'$ such that all the $\ell - 1$ edges on the path have valid edge configuration on them.*

Now we are ready to formulate the result that combines Bernshteyn's result [19] (equivalence between (1.) and (3.), and the moreover part) with the main results of this section. This also shows the remaining implications in Figure 1.

Theorem 16. *Let Π be an LCL on regular trees. Then the following are equivalent:*

1. $\Pi \in \mathsf{BAIRE}$,
2. Π *admits a* TOAST *algorithm,*
3. Π *admits an ℓ-full set,*
4. $\Pi \in \mathsf{LOCAL}(O(\log(n)))$.

Moreover, any of the equivalent conditions is necessary for $\Pi \in \mathsf{fiid}$.

Next we discuss the proof of Theorem 16. We refer the reader to Bernshteyn's paper [19] for full proofs in the case of BAIRE and fiid, here we only sketch the argument for completeness. We also note that instead of using the toast construction, he used a path decomposition of acyclic graphs of Conley, Marks and Unger [40].

6.1 Sufficiency

We start by showing that the combinatorial condition is sufficient for BAIRE and $\mathsf{LOCAL}(O(\log(n)))$. Namely, it follows from the next results together with Proposition 12 that (2.) implies all the other conditions in Theorem 16. As discussed above the main idea is to color inductively along the decompositions.

Proposition 13. *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL that admits ℓ-full set $\mathcal{V}' \subseteq \mathcal{V}$ for some $\ell > 0$. Then Π admits a* TOAST *algorithm that produces a Π-coloring for every $(2\ell+2)$-toast $\mathcal{D}$.*

Proof sketch. Our aim is to build a partial extending function. Set $q := 2\ell + 2$. Let E be a piece in a q-toast $\mathcal{D}$ and suppose that $D_1, \dots, D_k \in \mathcal{D}$ are subsets of E such that the boundaries are separated. Suppose, moreover, that we have defined inductively a coloring of half-edges of vertices in $D = \bigcup D_i$ using only vertex configurations from $\mathcal{V}'$ such that every edge configuration $\mathcal{E}$ is satisfied for every edge in D.

We handle each connected component of $E \setminus D$ separately. Let A be one of them. Let $u \in A$ be a boundary vertex of E. Such an vertex exists since every vertex in E has degree Δ. The distance of u and any D_i is at least $2\ell + 2$ for every $i \in [k]$. We orient all the edges from A towards u. Moreover if $v_i \in A$ is a boundary vertex of some D_i we assign to v_i a path V_i of length ℓ towards u. Note that V_i and V_j have distance at least 1, in particular, are disjoint for $i \neq j \in [k]$. Now, until you encounter some path V_i, color any in manner half-edges of vertices in A inductively starting at u in such a way that edge configurations $\mathcal{E}$ are satisfied on every edge and only vertex configurations from $\mathcal{V}'$ are used. Use the definition of ℓ-full set to find a coloring of any such V_i and continue in a similar manner until the whole A is colored. □

Proposition 14 (ℓ-full $\Rightarrow \mathsf{LOCAL}(O(\log n))$)**.** *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL with an ℓ-full set $\mathcal{V}' \subseteq \mathcal{V}$. Then Π can be solved in $O(\log n)$ rounds in* LOCAL.

Proof. The proof uses a variant of the rake-and-compress decomposition considered in [35].

The Decomposition The decomposition is parameterized an integer $\ell' \geq 1$, and it decomposes the vertices of T into $2L-1$ layers

$$V = V_1^{\mathsf{R}} \cup V_1^{\mathsf{C}} \cup V_2^{\mathsf{R}} \cup V_2^{\mathsf{C}} \cup V_3^{\mathsf{R}} \cup V_3^{\mathsf{C}} \cup \cdots \cup V_L^{\mathsf{R}},$$

with $L = O(\log n)$. We write G_i^{C} to denote the subtree induced by the vertices $\left(\bigcup_{j=i+1}^{L} V_j^{\mathsf{R}}\right) \cup \left(\bigcup_{j=i}^{L-1} V_j^{\mathsf{C}}\right)$. Similarly, G_i^{R} is the subtree induced by the vertices $\left(\bigcup_{j=i}^{L} V_j^{\mathsf{R}}\right) \cup \left(\bigcup_{j=i}^{L-1} V_j^{\mathsf{C}}\right)$. The sets V_i^{R} and V_i^{C} are required to satisfy the following requirements.

- Each $v \in V_i^{\mathsf{R}}$ has degree at most one in the graph G_i^{R}.
- Each $v \in V_i^{\mathsf{C}}$ has degree exactly two in the graph G_i^{C}. Moreover, the V_i^{C}-vertices in G_i^{C} form paths with s vertices, with $\ell' \leq s \leq 2\ell'$.

For any given constant $\ell' \geq 1$, it was shown in [35] that such a decomposition of a tree T can be computed in $O(\log n)$ rounds. See Figure 8 for an example of such a decomposition with $\ell' = 4$.

The Algorithm Given such a decomposition with $\ell' = \max\{1, \ell-2\}$, Π can be solved in $O(\log n)$ rounds by labeling the vertices in this order: V_L^{R}, V_{L-1}^{C}, V_{L-1}^{R}, …, V_1^{R}, as follows. The algorithm only uses the vertex configurations in the ℓ-full set $\mathcal{V}'$.

Labeling V_i^{R} Suppose all vertices in $V_L^{\mathsf{R}}, V_{L-1}^{\mathsf{C}}, V_{L-1}^{\mathsf{R}}, \ldots, V_i^{\mathsf{C}}$ have been labeled using $\mathcal{V}'$. Recall that each $v \in V_i^{\mathsf{R}}$ has degree at most one in the graph G_i^{R}. If $v \in V_i^{\mathsf{R}}$ has no neighbor in $V_L^{\mathsf{R}} \cup V_{L-1}^{\mathsf{C}} \cup V_{L-1}^{\mathsf{R}} \cup \cdots \cup V_i^{\mathsf{C}}$, then we can label the half edges surrounding v by any $c \in \mathcal{V}'$. Otherwise, $v \in V_i^{\mathsf{R}}$ has exactly one neighbor u in $V_L^{\mathsf{R}} \cup V_{L-1}^{\mathsf{C}} \cup V_{L-1}^{\mathsf{R}} \cup \cdots \cup V_i^{\mathsf{C}}$. Suppose the vertex configuration of u is c, where the half-edge label on $\{u, v\}$ is $\mathsf{a} \in c$. A simple observation from the definition of ℓ-full sets is that for any $c \in \mathcal{V}'$ and any $\mathsf{a} \in c$, there exist $c' \in \mathcal{V}'$ and $\mathsf{a}' \in c'$ in such a way that $\{\mathsf{a}, \mathsf{a}'\} \in \mathcal{E}$. Hence we can label the half edges surrounding v by $c' \in \mathcal{V}'$ where the half-edge label on $\{u, v\}$ is $\mathsf{a}' \in c'$.

Labeling V_i^{C} Suppose all vertices in $V_L^{\mathsf{R}}, V_{L-1}^{\mathsf{C}}, V_{L-1}^{\mathsf{R}}, \ldots, V_{i+1}^{\mathsf{R}}$ have been labeled using $\mathcal{V}'$. Recall that the V_i^{C}-vertices in G_i^{C} form degree-2 paths $P = (v_1, v_2, \ldots, v_s)$, with $\ell' \leq s \leq 2\ell'$. Let $P' = (x, v_1, v_2, \ldots, v_s, y)$ be the path resulting from appending to P the neighbors of the two endpoints of P in G_i^{C}. The two vertices x and y are in $V_L^{\mathsf{R}} \cup V_{L-1}^{\mathsf{C}} \cup V_{L-1}^{\mathsf{R}} \cup \cdots \cup V_{i+1}^{\mathsf{R}}$, so they have been assigned half-edge labels using $\mathcal{V}'$. Since P' contains at least $\ell' + 2 \geq \ell$ vertices, the definition of ℓ-full sets ensures that we can label $v_1, v_2, \ldots, v_s$ using vertex configurations in $\mathcal{V}'$ in such a way that the half-edge labels on $\{x, v_1\}, \{v_1, v_2\}, \ldots, \{v_s, y\}$ are all in $\mathcal{E}$. □

6.2 Necessity

We start by sketching that (2.) in Theorem 16 is necessary for BAIRE and fiid.

Theorem 17 (Bernshteyn [19])**.** *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL and suppose that $\Pi \in$ BAIRE or $\Pi \in$ fiid. Then Π admits an ℓ-full set $\mathcal{V}' \subseteq \mathcal{V}$ for some $\ell > 0$.*

Proof Sketch. We start with BAIRE. Suppose that every Borel acyclic Δ-regular graph admits a Borel solution on a τ-comeager set for every compatible Polish topology τ. In particular, this holds for the Borel graph induced by the standard generators of the free product of Δ-copies of $\mathbb{Z}_2$ on

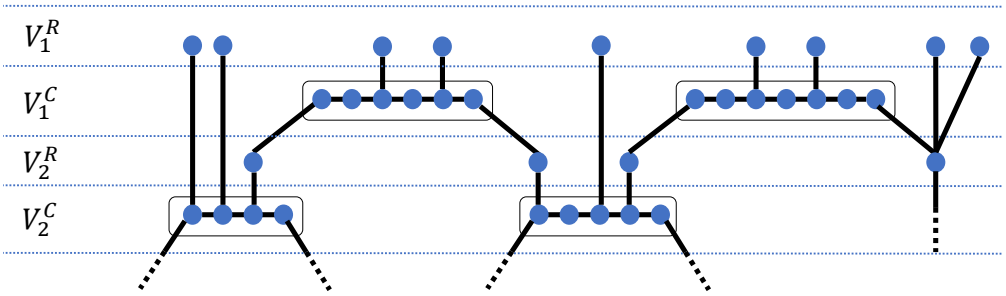

Figure 8: The variant of the rake-and-compress decomposition used in the proof of Proposition 14.

the free part of the shift action on the alphabet $\{0,1\}$ endowed with the product topology. Let F be such a solution. Write $\mathcal{V}' \subseteq \mathcal{V}$ for the configurations of half-edge labels around vertices that F outputs on a non-meager set. Let C be a comeager set on which F is continuous. Then, every element of $\mathcal{V}'$ is encoded by some finite window in the shift on C, that is, for each element there are a $k \in \mathbb{N}$ and function $s : B(1,k) \to \{0,1\}$ such that F is constant on the set $N_s \cap C$ (where N_s is the basic open neighbourhood determined by s, and $B(1,k)$ is the k-neighbourhood of the identity in the Cayley graph of the group). Since $\mathcal{V}'$ is finite, we can take $t > 0$ to be the maximum of such k's. It follows by standard arguments that $\mathcal{V}'$ is ℓ-full for $\ell > 2t + 1$.

A similar argument works for the fiid, however, for the sake of brevity, we sketch a shorter argument that uses the fact that there must be a correlation decay for factors of iid's. Let $\Pi \in \mathsf{fiid}$. That is, there is an $\mathrm{Aut}(T)$-equivariant measurable function from iid's on T (without colored edges this time) into the space of Π-colorings. Let $\mathcal{V}'$ be the set of half-edges configurations around vertices that have non-zero probability to appear. Let $u, v \in T$ be vertices of distance $k_0 \in \mathbb{N}$. By [6] the correlation between the configurations around u and v tends to 0 as $k_0 \to \infty$. This means that if the distance is big enough, then all possible pairs of $\mathcal{V}'$ configurations need to appear. □

To finish the proof of Theorem 16 we need to demonstrate the following theorem. Note that $\mathsf{LOCAL}(n^{o(1)}) = \mathsf{LOCAL}(O(\log n))$ according to the $\omega(\log n)$ – $n^{o(1)}$ complexity gap [35].

Theorem 18. *Let $\Pi = (\Sigma, \mathcal{V}, \mathcal{E})$ be an LCL solvable in $\mathsf{LOCAL}(n^{o(1)})$ rounds. Then there exists an ℓ-full set $\mathcal{V}' \subseteq \mathcal{V}$ for some $\ell \geq 2$.*

The rest of the section is devoted to the proof of Theorem 18. We start with the high-level idea of the proof. A natural attempt for showing $\mathsf{LOCAL}(n^{o(1)}) \Rightarrow \ell$-full is to simply take any $\mathsf{LOCAL}(n^{o(1)})$ algorithm $\mathcal{A}$ solving Π, and then take $\mathcal{V}'$ to be all vertex configurations that can possibly occur in an output of $\mathcal{A}$. It is not hard to see that this approach does not work in general, because the algorithm might use a special strategy to label vertices with degree smaller than Δ. Specifically, there might be some vertex configuration c used by $\mathcal{A}$ so that some $\mathsf{a} \in c$ will only be used to label *virtual* half edges. It will be problematic to include c in $\mathcal{V}'$.

To cope with this issue, we do not deal with general bounded-degree trees. Instead, we construct recursively a sequence $(W_1^*, W_2^*, \ldots, W_L^*)$ of sets of rooted, layered, and *partially labeled* tree in a special manner. A tree T is included in W_i^* if it can be constructed by gluing a multiset of rooted trees in W_{i-1}^* and a new root vertex r in a certain fixed manner. A vertex is said to be in layer i if it is introduced during the i-th step of the construction, i.e., it is introduced as the root r during the construction of W_i^* from W_{i-1}^*. All remaining vertices are said to be in layer 0.

We show that each $T \in W_L^*$ admits a correct labeling that extends the given partial labeling, as these partial labelings are computed by a simulation of $\mathcal{A}$. Moreover, in these correct labelings,

the variety of possible configurations of half-edge labels around vertices in different non-zero layers is the same for each layer. This includes vertices of non-zero layer whose half-edges are labeled by the given partial labeling. We simply pick $\mathcal{V}'$ to be the set of all configurations of half-edge labels around vertices that can appear in a non-zero layer in a correct labeling of a tree $T \in W_L^*$. Our construction ensures that each $c \in \mathcal{V}'$ appears as the labeling of some degree-Δ vertex in some tree that we consider.

The proof that $\mathcal{V}'$ is an ℓ-full set is based on finding paths using vertices of non-zero layers connecting two vertices with any two vertex configurations in $\mathcal{V}'$ in different lengths. These paths exist because the way rooted trees in W_{i-1}^* are glued together in the construction of W_i^* is sufficiently flexible. The reason that we need $\mathcal{A}$ to have complexity $\mathsf{LOCAL}(n^{o(1)})$ is that the construction of the trees can be parameterized by a number w so that all the trees have size polynomial in w and the vertices needed to be assigned labeling are at least distance w apart from each other. Since the number of rounds of $\mathcal{A}$ executed on trees of size $w^{O(1)}$ is much less than w, each labeling assignment can be calculated locally and independently. The construction of the trees as well as the analysis are based on a machinery developed in [35]. Specifically, we will consider the equivalence relation $\overset{\star}{\sim}$ defined in [35] and prove some of its properties, including a pumping lemma for bipolar trees. The exact definition of $\overset{\star}{\sim}$ in this paper is different from the one in [35] because the mathematical formalism describing LCL problems in this paper is different from the one in [35]. After that, we will consider a procedure for gluing trees parameterized by a labeling function f similar to the one used in [35]. We will apply this procedure iteratively to generate a set of trees. We will show that the desired ℓ-full set $\mathcal{V}' \subseteq \mathcal{V}$ can be constructed by considering the set of all possible correct labeling of these trees.

The Equivalence Relation $\overset{\star}{\sim}$ We consider trees with a list of designated vertices $v_1, v_2, \ldots, v_k$ called *poles*. A *rooted tree* is a tree with one pole r, and a *bipolar tree* is a tree with two poles s and t. For a tree T with its poles $S = (v_1, v_2, \ldots, v_k)$ with $\deg(v_i) = d_i < \Delta$, we denote by $h_{(T,S)}$ the function that maps each choice of the *virtual* half-edge labeling surrounding the poles of T to YES or NO, indicating whether such a partial labeling can be completed into a correct complete labeling of T. More specifically, consider

$$X = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k),$$

where $\mathcal{I}_i$ is a size-$(\Delta - d_i)$ multiset of labels in Σ, for each $1 \le i \le k$. Then $h_{(T,S)}(X) = \mathsf{YES}$ if there is a correct labeling of T such that the $\Delta - d_i$ virtual half-edge labels surrounding v_i are labeled by $\mathcal{I}_i$, for each $1 \le i \le k$. This definition can be generalized to the case T is already partially labeled in the sense that some of the half-edge labels have been fixed. In this case, $h_{(T,S)}(X) = \mathsf{NO}$ whenever X is incompatible with the given partial labeling.

Let T be a tree with poles $S = (v_1, v_2, \ldots, v_k)$ and let T' be another tree with poles $S' = (v'_1, v'_2, \ldots, v'_k)$ such that $\deg(v_i) = \deg(v'_i) = d_i$ for each $1 \le i \le k$. Then we write $T_1 \overset{\star}{\sim} T_2$ if $h_{(T,S)} = h_{(T',S')}$.

Given an LCL problem Π, it is clear that the number of equivalence classes of rooted trees and bipolar trees w.r.t. $\overset{\star}{\sim}$ is finite. For a rooted tree T, denote by $\mathsf{Class}_1(T)$ the equivalence class of T. For a bipolar tree H, denote by $\mathsf{Class}_2(H)$ the equivalence class of H.

Subtree Replacement The following lemma provides a sufficient condition that the equivalence class of a tree T is invariant of the equivalence class of its subtree T'. We note that a real half edge in T might become virtual in its subtree T'. Consider a vertex v in T' and its neighbor u that is in T but not in T'. Then the half edge $(v, \{u, v\})$ is real in T and virtual in T'.

Lemma 6 (Replacing subtrees)**.** *Let T be a tree with poles S. Let T' be a connected subtree of T induced by $U \subseteq V$, where V is the set of vertices in T. We identify a list of designated vertices $S' = (v'_1, v'_2, \ldots, v'_k)$ in U satisfying the following two conditions to be the poles of T'.*

- *$S \cap U \subseteq S'$.*
- *Each edge $e = \{u, v\}$ connecting $u \in U$ and $v \in V \setminus U$ must satisfy $u \in S'$.*

Let T'' be another tree with poles $S'' = (v''_1, v''_2, \ldots, v''_k)$ that is in the same equivalence class as T'. Let T^* be the result of replacing T' by T'' in T by identifying $v'_i = v''_i$ for each $1 \leq i \leq k$. Then T^* is in the same equivalence class as T.

Proof. To prove the lemma, by symmetry, it suffices to show that starting from any correct labeling $\mathcal{L}$ of T, it is possible to find a correct labeling $\mathcal{L}^*$ of T^* in such a way that the multiset of the virtual half-edge labels surrounding each pole in S remain the same.

Such a correct labeling $\mathcal{L}^*$ of T^* is constructed as follows. If $v \in V \setminus U$, then we simply adopt the given labeling $\mathcal{L}$ of v in T. Next, consider the vertices in U. Set $X = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$ to be the one compatible with the labeling $\mathcal{L}$ of T restricted to the subtree T' in the sense that $\mathcal{I}_i$ is the multiset of the virtual half-edge labels surrounding the pole v'_i of T', for each $1 \leq i \leq k$. We must have $h_{T',S'}(X) = \mathsf{YES}$. Since T' and T'' are in the same equivalence class, we have $h_{T'',S''}(X) = \mathsf{YES}$, and so we can find a correct labeling $\mathcal{L}''$ of T'' that is also compatible with X. Combining this labeling $\mathcal{L}''$ of the vertices U in T'' with the labeling $\mathcal{L}$ of the vertices $V \setminus U$ in T, we obtain a desired labeling $\mathcal{L}^*$ of T^*.

We verify that $\mathcal{L}^*$ gives a correct labeling of T^*. Clearly, the size-Δ multiset that labels each vertex $v \in V$ is in $\mathcal{V}$, by the correctness of $\mathcal{L}$ and $\mathcal{L}''$. Consider any edge $e = \{u, v\}$ in T^*. Similarly, if $\{u, v\} \subseteq V \setminus U$ or $\{u, v\} \cap (V \setminus U) = \emptyset$, then the size-2 multiset that labels e is in $\mathcal{E}$, by the correctness of $\mathcal{L}$ and $\mathcal{L}''$. For the case that $e = \{u, v\}$ connects a vertex $u \notin V$ and a vertex $v \in V \setminus U$, we must have $u \in S'$ by the lemma statement. Therefore, the label of the half edge (u, e) is the same in both $\mathcal{L}$ and $\mathcal{L}^*$ by our choice of $\mathcal{L}''$. Thus, the size-2 multiset that labels e is in $\mathcal{E}$, by the correctness of $\mathcal{L}$.

We verify that the virtual half-edge labels surrounding each pole in S are the same in both $\mathcal{L}$ and $\mathcal{L}^*$. Consider a pole $v \in S$. If $v \in V \setminus U$, then the labeling of v is clearly the same in both $\mathcal{L}$ and $\mathcal{L}^*$. If $v \notin V \setminus U$, then the condition $S \cap U \subseteq S'$ in the statement implies that $v \in S'$. In this case, the way we pick $\mathcal{L}''$ ensures that the virtual half-edge labels surrounding $v \in S'$ are the same in both $\mathcal{L}$ and $\mathcal{L}^*$. □

In view of the proof of Lemma 6, as long as the conditions in Lemma 6 are met, we are able to abstract out a subtree by its equivalence class when reasoning about correct labelings of a tree. This observation will be applied repeatedly in the subsequent discussion.

A Pumping Lemma We will prove a pumping lemma of bipolar trees using Lemma 6. Suppose T_i is a tree with a root r_i for each $1 \leq i \leq k$, then $H = (T_1, T_2, \ldots, T_k)$ denotes the bipolar tree resulting from concatenating the roots $r_1, r_2, \ldots, r_k$ into a path $(r_1, r_2, \ldots, r_k)$ and setting the two poles of H by $s = r_1$ and $t = r_k$. A simple consequence of Lemma 6 is that $\mathsf{Class}_2(H)$ is determined by $\mathsf{Class}_1(T_1), \mathsf{Class}_1(T_2), \ldots, \mathsf{Class}_1(T_k)$. We have the following pumping lemma.

Lemma 7 (Pumping lemma)**.** *There exists a finite number $\ell_{\text{pump}} > 0$ such that as long as $k \geq \ell_{\text{pump}}$, any bipolar tree $H = (T_1, T_2, \ldots, T_k)$ can be decomposed into $H = X \circ Y \circ Z$ with $0 < |Y| < k$ so that $X \circ Y^i \circ Z$ is in the same equivalence class as H for each $i \geq 0$.*

Proof. Set ℓ_{pump} to be the number of equivalence classes for bipolar trees plus one. By the pigeon hole principle, there exist $1 \le a < b \le k$ such that $(T_1, T_2, \ldots, T_a)$ and $(T_1, T_2, \ldots, T_b)$ are in the same equivalence class. Set $X = (T_1, T_2, \ldots, T_a)$, $Y = (T_{a+1}, T_{a+2}, \ldots, T_b)$, and $Z = (T_{b+1}, T_{b+2}, \ldots, T_k)$. As we already know that $\mathsf{Class}_2(X) = \mathsf{Class}_2(X \circ Y)$, Lemma 6 implies that $\mathsf{Class}_2(X \circ Y) = \mathsf{Class}_2(X^2 \circ Y)$ by replacing X by $X \circ Y$ in the bipolar tree $X \circ Y$. Similarly, $\mathsf{Class}_2(X \circ Y^i)$ is the same for for each $i \ge 0$. Applying Lemma 6 again to replace $X \circ Y$ by $X \circ Y^i$ in $H = X \circ Y \circ Z$, we conclude that $X \circ Y^i \circ Z$ is in the same equivalence class as H for each $i \ge 0$. □

A Procedure for Gluing Trees Suppose that we have a set of rooted trees W. We devise a procedure that generates a new set of rooted trees by gluing the rooted trees in W together. This procedure is parameterized by a labeling function f. Consider a bipolar tree

$$H = (T^l_1, T^l_2, \ldots, T^l_{\ell_{\text{pump}}}, T^m, T^r_1, T^r_2, \ldots, T^r_{\ell_{\text{pump}}})$$

where T^m is formed by attaching the roots $r_1, r_2, \ldots, r_{\Delta-2}$ of the rooted trees $T^m_1, T^m_2, \ldots, T^m_{\Delta-2}$ to the root r^m of T^m.

The labeling function f assigns the half-edge labels surrounding r^m based on

$$\mathsf{Class}_2(H^l), \mathsf{Class}_1(T^m_1), \mathsf{Class}_1(T^m_2), \ldots, \mathsf{Class}_1(T^m_{\Delta-2}), \mathsf{Class}_2(H^r)$$

where $H^l = (T^l_1, T^l_2, \ldots, T^l_{\ell_{\text{pump}}})$ and $H^r = (T^r_1, T^r_2, \ldots, T^r_{\ell_{\text{pump}}})$.

We write T^m_* to denote the result of applying f to label the root r^m of T^m in the bipolar tree H, and we write

$$H_* = (T^l_1, T^l_2, \ldots, T^l_{\ell_{\text{pump}}}, T^m_*, T^r_1, T^r_2, \ldots, T^r_{\ell_{\text{pump}}})$$

to denote the result of applying f to H. We make the following observation.

Lemma 8 (Property of f)**.** *The two equivalence classes $\mathsf{Class}_1(T^m_*)$ and $\mathsf{Class}_2(H_*)$ are determined by*

$$\mathsf{Class}_2(H^l), \mathsf{Class}_1(T^m_1), \mathsf{Class}_1(T^m_2), \ldots, \mathsf{Class}_1(T^m_{\Delta-2}), \mathsf{Class}_2(H^r),$$

and the labeling function f.

Proof. By Lemma 6, once the half-edge labelings of the root r^m of T^m is fixed, $\mathsf{Class}_1(T^m_*)$ is determined by $\mathsf{Class}_1(T^m_1), \mathsf{Class}_1(T^m_2), \ldots, \mathsf{Class}_1(T^m_{\Delta-2})$. Therefore, indeed $\mathsf{Class}_1(T^m_*)$ is determined by

$$\mathsf{Class}_2(H^l), \mathsf{Class}_1(T^m_1), \mathsf{Class}_1(T^m_2), \ldots, \mathsf{Class}_1(T^m_{\Delta-2}), \mathsf{Class}_2(H^r),$$

and the labeling function f. Similarly, applying Lemma 6 to the decomposition of H_* into H^l, T^m_*, and H^r, we infer that $\mathsf{Class}_2(H_*)$ depends only on $\mathsf{Class}_2(H^l)$, $\mathsf{Class}_1(T^m_*)$, and $\mathsf{Class}_2(H^r)$. □

The three sets of trees $X_f(W)$, $Y_f(W)$, and $Z_f(W)$ are constructed as follows.

- $X_f(W)$ is the set of all rooted trees resulting from appending $\Delta - 2$ arbitrary rooted trees $T_1, T_2, \ldots, T_{\Delta-2}$ in W to a new root vertex r.
- $Y_f(W)$ is the set of bipolar trees constructed as follows. For each choice of $2\ell_{\text{pump}} + 1$ rooted trees $T^l_1, T^l_2, \ldots, T^l_{\ell_{\text{pump}}}, T^m, T^r_1, T^r_2, \ldots, T^r_{\ell_{\text{pump}}}$ from $X_f(W)$, concatenate them into a bipolar tree
$$H = (T^l_1, T^l_2, \ldots, T^l_{\ell_{\text{pump}}}, T^m, T^r_1, T^r_2, \ldots, T^r_{\ell_{\text{pump}}}),$$
let H_* be the result of applying the labeling function f to H, and then add H_* to $Y_f(W)$.

- $Z_f(W)$ is the set of rooted trees constructed as follows. For each

$$H_* = (T_1^l, T_2^l, \ldots, T_{\ell_{\text{pump}}}^l, T_*^m, T_1^r, T_2^r, \ldots, T_{\ell_{\text{pump}}}^r) \in Y_f(W),$$

  add $(T_1^l, T_2^l, \ldots, T_{\ell_{\text{pump}}}^l, T_*^m)$ to $Z_f(W)$, where we set the root of T_1^l as the root, and add $(T_*^m, T_1^r, T_2^r, \ldots, T_{\ell_{\text{pump}}}^r)$ to $Z_f(W)$, where we set the root of $T_{\ell_{\text{pump}}}^r$ as the root.

We write $\mathsf{Class}_1(S) = \bigcup_{T \in S}\{\mathsf{Class}_1(T)\}$ and $\mathsf{Class}_2(S) = \bigcup_{H \in S}\{\mathsf{Class}_2(H)\}$, and we make the following observation.

Lemma 9 (Property of $X_f(W)$, $Y_f(W)$, and $Z_f(W)$)**.** *The three sets of equivalence classes $\mathsf{Class}_1(X_f(W))$, $\mathsf{Class}_2(Y_f(W))$, and $\mathsf{Class}_1(Z_f(W))$ depend only on $\mathsf{Class}_1(W)$ and the labeling function f.*

Proof. This is a simple consequence of Lemmas 6 and 8. □

A Fixed Point W^* Given a fixed labeling function f, we want to find a set of rooted trees W^* that is a *fixed point* for the procedure Z_f in the sense that

$$\mathsf{Class}_1(Z_f(W^*)) = \mathsf{Class}_1(W^*).$$

To find such a set W^*, we construct a two-dimensional array of rooted trees $\{W_{i,j}\}$, as follows.

- For the base case, $W_{1,1}$ consists of only the one-vertex rooted tree.
- Given that $W_{i,j}$ as been constructed, we define $W_{i,j+1} = Z_f(W_{i,j})$.
- Given that $W_{i,j}$ for all positive integers j have been constructed, $W_{i+1,1}$ is defined as follows. Pick b_i as the smallest index such that $\mathsf{Class}_1(W_{i,b_i}) = \mathsf{Class}_1(W_{i,a_i})$ for some $1 \le a_i < b_i$. By the pigeon hole principle, the index b_i exists, and it is upper bounded by 2^C, where C is the number of equivalence classes for rooted trees. We set

$$W_{i+1,1} = W_{i,a_i} \cup W_{i,a_i+1} \cup \cdots \cup W_{i,b_i-1}.$$

We show that the sequence $\mathsf{Class}_1(W_{i,a_i}), \mathsf{Class}_1(W_{i,a_i+1}), \mathsf{Class}_1(W_{i,a_i+2}), \ldots$ is periodic with a period $c_i = b_i - a_i$.

Lemma 10. *For any $i \ge 1$ and for any $j \ge a_i$, we have $\mathsf{Class}_1(W_{i,j}) = \mathsf{Class}_1(W_{i,j+c_i})$, where $c_i = b_i - a_i$.*

Proof. By Lemma 9, $\mathsf{Class}_1(W_{i,j})$ depends only on $\mathsf{Class}_1(W_{i,j-1})$. Hence the lemma follows from the fact that $\mathsf{Class}_1(W_{i,b_i}) = \mathsf{Class}_1(W_{i,a_i})$. □

Next, we show that $\mathsf{Class}_1(W_{2,1}) \subseteq \mathsf{Class}_1(W_{3,1}) \subseteq \mathsf{Class}_1(W_{4,1}) \subseteq \cdots$.

Lemma 11. *For any $i \ge 2$, we have $\mathsf{Class}_1(W_{i,1}) \subseteq \mathsf{Class}_1(W_{i,j})$ for each $j > 1$, and so $\mathsf{Class}_1(W_{i,1}) \subseteq \mathsf{Class}_1(W_{i+1,1})$.*

Proof. Since $W_{i,1} = \bigcup_{a_{i-1} \le l \le b_{i-1}-1} W_{i-1,l}$, we have

$$\bigcup_{a_{i-1}+j-1 \le l \le b_{i-1}+j} W_{i-1,l} \subseteq W_{i,j}$$

according to the procedure of constructing $W_{i,j}$. By Lemma 10,

$$\mathsf{Class}_1\left(\bigcup_{a_{i-1}\le l\le b_{i-1}-1} W_{i-1,l}\right) = \mathsf{Class}_1\left(\bigcup_{a_{i-1}+j-1\le l\le b_{i-1}+j} W_{i-1,l}\right)$$

for all $j \ge 1$, and so we have $\mathsf{Class}_1(W_{i,1}) \subseteq \mathsf{Class}_1(W_{i,j})$ for each $j > 1$. The claim $\mathsf{Class}_1(W_{i,1}) \subseteq \mathsf{Class}_1(W_{i+1,1})$ follows from the fact that $W_{i+1,1} = \bigcup_{a_i\le l\le b_i-1} W_{i,l}$. □

Set i^* to be the smallest index $i \ge 2$ such that $\mathsf{Class}_1(W_{i,1}) = \mathsf{Class}_1(W_{i+1,1})$. By the pigeon hole principle and Lemma 11, the index i^* exists, and it is upper bounded by C, the number of equivalence classes for rooted trees. We set

$$W^* = W_{i^*,1}.$$

The following lemma shows that $\mathsf{Class}_1(Z_f(W^*)) = \mathsf{Class}_1(W^*)$, as needed.

Lemma 12. *For any $i \ge 2$, if $\mathsf{Class}_1(W_{i,1}) = \mathsf{Class}_1(W_{i+1,1})$, then $\mathsf{Class}_1(W_{i,j})$ is the same for all $j \ge 1$.*

Proof. By Lemma 10 and the way we construct $W_{i+1,1}$, we have

$$\mathsf{Class}_1(W_{i,j}) \subseteq \mathsf{Class}_1(W_{i+1,1}) \text{ for each } j \ge a_i.$$

By Lemma 11, we have

$$\mathsf{Class}_1(W_{i,1}) \subseteq \mathsf{Class}_1(W_{i,j}) \text{ for each } j > 1.$$

Therefore, $\mathsf{Class}_1(W_{i,1}) = \mathsf{Class}_1(W_{i+1,1})$ implies that $\mathsf{Class}_1(W_{i,j}) = \mathsf{Class}_1(W_{i,1})$ for each $j \ge a_i$. Hence we must have $\mathsf{Class}_1(Z_f(W_{i,1})) = \mathsf{Class}_1(W_{i,1})$, and so $\mathsf{Class}_1(W_{i,j})$ is the same for all $j \ge 1$. □

We remark that simply selecting W^* to be any set such that $\mathsf{Class}_1(Z_f(W^*)) = \mathsf{Class}_1(W^*)$ is not enough for our purpose. As we will later see, it is crucial that the set W^* is constructed by iteratively applying the function Z_f and taking the union of previously constructed sets.

A Sequence of Sets of Trees We define $W_1^* = W^*$ and $W_i^* = Z_f(W_{i-1}^*)$ for each $1 < i \le L$, where L is some sufficiently large number to be determined.

The way we choose W^* guarantees that $\mathsf{Class}_1(W_i^*) = \mathsf{Class}_1(W^*)$ for all $1 \le i \le L$. For convenience, we write $X_i^* = X_f(W_i^*)$ and $Y_i^* = Y_f(W_i^*)$. Similarly, $\mathsf{Class}_1(X_i^*)$ is the same for all $1 \le i \le L$ and $\mathsf{Class}_2(Y_i^*)$ is the same for all $1 \le i \le L$.

Our analysis will rely on the assumption that all rooted trees in W^* admit correct labelings. Whether this is true depends only on $\mathsf{Class}_1(W^*)$, which depends only on the labeling function f. We say that f is *feasible* if it leads to a set W^* where all the rooted trees therein admit correct labelings. The proof that a feasible labeling function f exists is deferred.

The assumption that f is feasible implies that all trees in W_i^*, X_i^*, and Y_i^*, for all $1 \le i \le L$, admit correct labelings. All rooted trees in X_i^* admit correct labelings because they are subtrees of the rooted trees in W_{i+1}^*. A correct labeling of any bipolar tree $H \in Y_i^*$ can be obtained by combining any correct labelings of the two rooted trees in W_{i+1}^* resulting from H.

Layers of Vertices We assign a layer number $\lambda(v)$ to each vertex v in a tree based on the step that v is introduced in the construction

$$W_1^* \to W_2^* \to \cdots \to W_L^*.$$

If a vertex v is introduced as the root vertex of a tree in $X_i^* = X_f(W_i^*)$, then we say that the layer number of v is $\lambda(v) = i \in \{1, 2, \ldots, L\}$. A vertex v has $\lambda(v) = 0$ if it belongs to a tree in W_1^*.

For any vertex v with $\lambda(v) = i$ in a tree $T \in X_j^*$ with $i \le j$, we write T_v to denote the subtree of T such that $T_v \in X_i^*$ where v is the root of T_v.

We construct a sequence of sets $R_1, R_2, \ldots, R_L$ as follows. We go over all rooted trees $T \in X_L^*$, all possible correct labeling $\mathcal{L}$ of T, and all vertices v in T with $\lambda(v) = i \in \{1, 2, \ldots, L\}$. Suppose that the $\Delta - 2$ rooted trees in the construction of $T_v \in X_i^* = X_f(W_i^*)$ are T_1, T_2, …, $T_{\Delta-2}$, and let r_i be the root of T_i. Consider the following parameters.

- $c_i = \mathsf{Class}_1(T_i)$.
- $\mathtt{a}_i$ is the real half-edge label of v in T_v for the edge $\{v, r_i\}$, under the correct labeling $\mathcal{L}$ of T restricted to T_v.
- $\mathcal{I}$ is the size-2 multiset of virtual half-edge labels of v in T_v, under the correct labeling $\mathcal{L}$ of T restricted to T_v.

Then we add $(c_1, c_2, \ldots, c_{\Delta-2}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}, \mathcal{I})$ to R_i.

Lemma 13. *For each $1 \le i < L$, R_i is determined by R_{i+1}.*

Proof. We consider the following alternative way of constructing R_i from R_{i+1}. Each rooted tree T' in $W_{i+1}^* = Z_f(W_i^*)$ can be described as follows.

- Start with a path $(r_1, r_2, \ldots, r_{\ell_{\text{pump}}+1})$, where r_1 is the root of T'.
- For each $1 \le j \le \ell_{\text{pump}} + 1$, append $\Delta - 2$ rooted trees $T_{j,1}, T_{j,2}, \ldots, T_{j,\Delta-2} \in W_i^*$ to r_j.
- Assign the labels to the half edges surrounding $r_{\ell_{\text{pump}}+1}$ according to the labeling function f.

Now, consider the function ϕ that maps each equivalence class c for rooted trees to a subset of Σ defined as follows: $\mathtt{a} \in \phi(c)$ if there exist

$$(c_1, c_2, \ldots, c_{\Delta-2}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}, \mathcal{I}) \in R_{i+1}$$

and $1 \le j \le \Delta - 2$ such that $c = c_j$ and $\mathtt{a} = \mathtt{a}_j$.

We go over all possible $T' \in W_{i+1}^* = Z_f(W_i^*)$. Note that the root r of T' has exactly one virtual half edge. For each $\mathtt{b} \in \Sigma$ such that $\{\mathtt{a}, \mathtt{b}\} \in \mathcal{E}$ for some $\mathtt{a} \in \phi(\mathsf{Class}_1(T'))$, we go over all possible correct labelings $\mathcal{L}$ of T' where the virtual half edge of r is labeled $\mathtt{b}$. For each $1 \le j \le \ell_{\text{pump}} + 1$, consider the following parameters.

- $c_l = \mathsf{Class}_1(T_{j,l})$.
- $\mathtt{a}_l$ is the half-edge label of r_j for the edge $\{r_j, r_{j,l}\}$.
- $\mathcal{I}$ is the size-2 multiset of the remaining two half-edge labels of v.

Then we add $(c_1, c_2, \ldots, c_{\Delta-2}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}, \mathcal{I})$ to R_i.

This construction of R_i is equivalent to the original construction of R_i because a correct labeling of $T' \in W_i^*$ can be extended to a correct labeling of a tree $T \in X_L^*$ that contains T' as a subtree if and only if the virtual half-edge label of the root r of T' is $\mathtt{b} \in \Sigma$ such that $\{\mathtt{a}, \mathtt{b}\} \in \mathcal{E}$ for some $\mathtt{a} \in \phi(\mathsf{Class}_1(T'))$.

It is clear that this construction of R_i only depends on $\mathsf{Class}_1(W_i^*)$, the labeling function f, and the function ϕ, which depends only on R_{i+1}. Since the labeling function f is fixed and $\mathsf{Class}_1(W_i^*)$ is the same for all i, we conclude that R_i depends only on R_{i+1}. □

Lemma 14. *We have $R_1 \subseteq R_2 \subseteq \cdots \subseteq R_L$.*

Proof. For the base case, we show that $R_{L-1} \subseteq R_L$. In fact, our proof will show that $R_i \subseteq R_L$ for each $1 \le i < L$. Consider any

$$(c_1, c_2, \ldots, c_{\Delta-2}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}, \mathcal{I}) \in R_i.$$

Then there is a rooted tree $T \in X_i^*$ that is formed by attaching $\Delta - 2$ rooted trees of equivalence classes $c_1, c_2, \ldots, c_{\Delta-2}$ to the root vertex so that if we label the half edges surrounding the root vertex according to $\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}$, and $\mathcal{I}$, then this partial labeling can be completed into a correct labeling of T.

Because $\mathsf{Class}_1(W_i^*) = \mathsf{Class}_1(W_L^*)$, there is also a rooted tree $T' \in X_L^*$ that is formed by attaching $\Delta - 2$ rooted trees of equivalence classes $c_1, c_2, \ldots, c_{\Delta-2}$ to the root vertex. Therefore, if we label the root vertex of T' in the same way as we do for T, then this partial labeling can also be completed into a correct labeling of T'. Hence we must have

$$(c_1, c_2, \ldots, c_{\Delta-2}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}, \mathcal{I}) \in R_L.$$

Now, suppose that we already have $R_i \subseteq R_{i+1}$ for some $1 < i < L$. We will show that $R_{i-1} \subseteq R_i$. Denote by ϕ_i and ϕ_{i+1} the function ϕ in Lemma 13 constructed from R_i and R_{i+1}. We have $\phi_i(c) \subseteq \phi_{i+1}(c)$ for each equivalence class c, because $R_i \subseteq R_{i+1}$. Therefore, in view of the alternative construction described in the proof of Lemma 13, we have $R_{i-1} \subseteq R_i$. □

The Set of Vertex Configurations $\mathcal{V}'$ By Lemmas 13 and 14, if we pick L to be sufficient large, we can have $R_1 = R_2 = R_3$. More specifically, if we pick

$$L \ge C^{\Delta-2} \cdot \binom{|\Sigma|+1}{|\Sigma|-1} + 3,$$

then there exists an index $3 \le i \le L$ such that $R_i = R_{i-1}$, implying that $R_1 = R_2 = R_3$. Here C is the number of equivalence classes for rooted trees and $\binom{|\Sigma|+1}{|\Sigma|-1}$ is the number of size-2 multisets of elements from Σ.

The set $\mathcal{V}'$ is defined by including all size-Δ multisets $\{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}\} \cup \mathcal{I}$ such that

$$(c_1, c_2, \ldots, c_{\Delta-2}, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{\Delta-2}, \mathcal{I}) \in R_1$$

for some $c_1, c_2, \ldots, c_{\Delta-2}$.

The Set $\mathcal{V}'$ is ℓ-full To show that the set $\mathcal{V}'$ is ℓ-full, we consider the subset $\mathcal{V}^* \subseteq \mathcal{V}'$ defined by the set of vertex configurations used by the labeling function f in the construction of $Y_f(W_i^*)$. The definition of $\mathcal{V}^*$ is invariant of i as $\mathsf{Class}_1(W_i^*)$ is the same for all i. Clearly, for each $x \in \mathcal{V}^*$, there is a rooted tree $T \in W_2^*$ where the root of a subtree $T' \in X_1^*$ of T has its size-Δ multiset of half-edge labels fixed to be x by f, and so $\mathcal{V}^* \subseteq \mathcal{V}'$.

For notational simplicity, we write $x \overset{k}{\leftrightarrow} x'$ if there is a correct labeling of a k-vertex path $(v_1, v_2, \ldots, v_k)$ using only vertex configurations in $\mathcal{V}'$ so that the vertex configuration of v_1 is x and the vertex configuration of v_k is x'. If it is further required that the half-edge label of v_1 for the edge $\{v_1, v_2\}$ is $\mathtt{a} \in x$, then we write $(x, \mathtt{a}) \overset{k}{\leftrightarrow} x'$. The notation $(x, \mathtt{a}) \overset{k}{\leftrightarrow} (x', \mathtt{a}')$ is defined similarly.

Lemma 15. *For any $x \in \mathcal{V}' \setminus \mathcal{V}^*$ and $\mathtt{a} \in x$, there exist a vertex configuration $x' \in \mathcal{V}^*$ and a number $2 \le k \le 2\ell_{\text{pump}} + 1$ such that $(x, \mathtt{a}) \overset{k}{\leftrightarrow} x'$.*

Proof. Let $T \in W_L^*$ be chosen so that there is a correct labeling $\mathcal{L}$ where x is a vertex configuration of some vertex v with $\lambda(v) = 2$.

To prove the lemma, it suffices to show that for each of the Δ neighbors u of v, it is possible to find a path $P = (v, u, \ldots, w)$ meeting the following conditions.

- w is a vertex whose half-edge labels have been fixed by f. This ensures that the vertex configuration of w is in $\mathcal{V}^*$.
- All vertices in P are within layers 1,2, and 3. This ensures that the vertex configuration of all vertices in P are in $\mathcal{V}'$.
- The number of vertices k in P satisfies $2 \le k \le 2\ell_{\text{pump}} + 1$.

We divide the proof into three cases. Refer to Figure 9 for an illustration, where squares are vertices whose half-edge labels have been fixed by f.

Case 1 Consider the subtree $T_v \in X_2^*$ of T whose root is v. In view of the construction of the set $X_f(W_2^*)$, v has $\Delta - 2$ children $u_1, u_2, \ldots, u_{\Delta-2}$ in T_v, where the subtree T_i rooted at u_i is a rooted tree in W_2^*.

For each $1 \le i \le \Delta - 2$, according to the structure of the trees in the set $W_2^* = Z_f(W_1^*)$, there is a path $(u_i = w_1, w_2, \ldots, w_{\ell_{\text{pump}}+1})$ in T_i containing only layer-1 vertices, where the half-edge labels of $w_{\ell_{\text{pump}}+1}$ have been fixed by f. Hence $P = (v, u_i = w_1, w_2, \ldots, w_{\ell_{\text{pump}}+1})$ is a desired path with $k = \ell_{\text{pump}} + 2$ vertices.

Case 2 Consider the subtree $T' \in W_3^* = Z_f(W_2^*)$ that contains v in T. Similarly, according to the structure of the trees in the set $W_3^* = Z_f(W_2^*)$, there is a path $(r = w'_1, w'_2, \ldots, w'_{\ell_{\text{pump}}+1})$ in T' containing only layer-2 vertices so that r is the root of T', $v = w'_i$ for some $1 \le i' \le \ell_{\text{pump}} + 1$, and the half-edge labels of $w'_{\ell_{\text{pump}}+1}$ have been fixed by f. Since $x \in \mathcal{V}' \setminus \mathcal{V}^*$, we have $v \neq w'_{\ell_{\text{pump}}+1}$. Hence $P = (v = w'_i, w'_{i+1}, \ldots, w'_{\ell_{\text{pump}}+1})$ is a desired path with $2 \le k \le \ell_{\text{pump}} + 1$ vertices.

Case 3 There is only one remaining neighbor of v to consider. In view of the construction of $X_f(W_3^*)$, there is a layer-3 vertex v' adjacent to the vertex r, the root of the tree $T' \in W_3^*$ considered in the previous case. If the half-edge labels of v' have been fixed by f, then $P = (v = w'_i, w'_{i-1}, \ldots, w'_1 = r, v')$ is a desired path. Otherwise, similar to the analysis in the previous case, we can find a path $P' = (v', \ldots, w)$ connecting v' to a vertex w whose half-edge labels have been fixed by f. All vertices in P' are of layer-3, and the number of vertices in P' is within $[2, \ell_{\text{pump}} + 1]$.

Combining P' with the path $(v = w'_i, w'_{i-1}, \ldots, w'_1 = r, v')$, we obtain the desired path P whose number of vertices satisfies $2 \le k \le 2\ell_{\text{pump}} + 1$. □

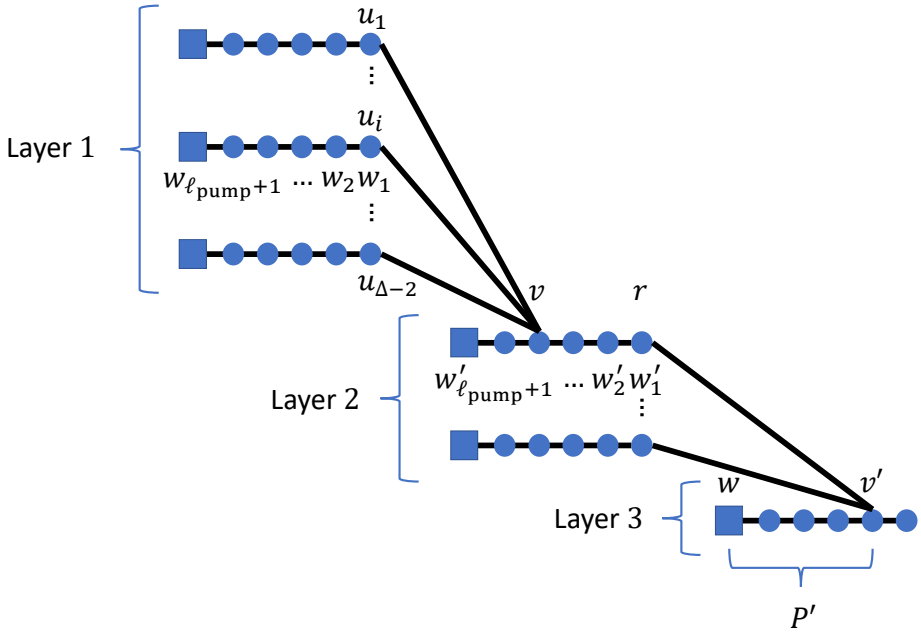

Figure 9: An illustration for the proof of Lemma 15.

For Lemma 16, note that if $\mathtt{a}$ appears more than once in the multiset x, then we still have $\mathtt{a} \in x \setminus \{\mathtt{a}\}$.

Lemma 16. *For any $\mathtt{a} \in x \in \mathcal{V}^*$, $\mathtt{a}' \in x' \in \mathcal{V}^*$, and $0 \le t \le \ell_{\text{pump}} - 1$, we have $(x, \mathtt{b}) \overset{k}{\leftrightarrow} (x', \mathtt{b}')$ for some $\mathtt{b} \in x \setminus \{\mathtt{a}\}$ and $\mathtt{b}' \in x' \setminus \{\mathtt{a}'\}$ with $k = 2\ell_{\text{pump}} + 4 + t$.*

Proof. For any $\mathtt{a} \in x \in \mathcal{V}^*$, we can find a rooted tree $T_{x,\mathtt{a}} \in W_2^*$ such that the path $(v_1, v_2, \ldots, v_{\ell_{\text{pump}}+1})$ of the layer-1 vertices in $T_{x,\mathtt{a}}$ where $v_1 = r$ is the root of $T_{x,\mathtt{a}}$ satisfies the property that the half-edge labels of $v_{\ell_{\text{pump}}+1}$ have been fixed to x by f where the half-edge label for $\{v_{\ell_{\text{pump}}}, v_{\ell_{\text{pump}}+1}\}$ is in $x \setminus \{\mathtt{a}\}$.

Now, observe that for any choice of $(\Delta - 2)(\ell_{\text{pump}} + 1)$ rooted trees

$$\{T_{i,j}\}_{1 \le i \le \ell_{\text{pump}}+1, 1 \le j \le \Delta-2}$$

in W_2^*, there is a rooted tree $T' \in W_3^* = Z_f(W_2^*)$ formed by appending $T_{i,1}, T_{i,2}, \ldots, T_{i,\Delta-2}$ to u_i in the path $(u_1, u_2, \ldots, u_{\ell_{\text{pump}}+1})$ and fixing the half-edge labeling of $u_{\ell_{\text{pump}}+1}$ by f. All vertices in $(u_1, u_2, \ldots, u_{\ell_{\text{pump}}+1})$ are of layer-2.

We choose any $T' \in W_2^*$ with $T_{i,j} = T_{x,\mathtt{a}}$ and $T_{i',j'} = T_{x',\mathtt{a}'}$ such that $i' - i = t + 1$. The possible range of t is $[0, \ell_{\text{pump}} - 1]$. Consider any $T \in W_L^*$ that contains T' as its subtree, and consider any correct labeling of T. Then the path P resulting from concatenating the path $(v_{\ell_{\text{pump}}+1}, v_{\ell_{\text{pump}}}, \ldots, v_1)$ in $T_{x,\mathtt{a}}$, the path $(u_i, u_{i+1}, \ldots, u_{i'})$, and the path $(v'_1, v'_2, \ldots, v'_{\ell_{\text{pump}}+1})$ in $T_{x',\mathtt{a}'}$ shows that $(x, \mathtt{b}) \overset{k}{\leftrightarrow} (x', \mathtt{b}')$ for $k = (\ell_{\text{pump}} + 1) + (t + 2) + (\ell_{\text{pump}} + 1) = 2\ell_{\text{pump}} + 4 + t$. See Figure 10 for an illustration.

For the rest of the proof, we show that the desired rooted tree $T_{x,\mathtt{a}} \in W_2^*$ exists for any $\mathtt{a} \in x \in \mathcal{V}^*$. For any $x \in \mathcal{V}^*$, we can find a bipolar tree

$$H_* = (T_1^l, T_2^l, \ldots, T_{\ell_{\text{pump}}}^l, T_*^m, T_1^r, T_2^r, \ldots, T_{\ell_{\text{pump}}}^r) \in Y_1^* = Y_f(W_1^*)$$

such that the vertex configuration of the root r^m of T_*^m is fixed to be x by the labeling function f. Then, for any $\mathtt{a} \in x$, at least one of the two rooted trees in $W_2^* = Z_f(W_1^*)$ resulting from cutting H_* satisfies the desired requirement. □

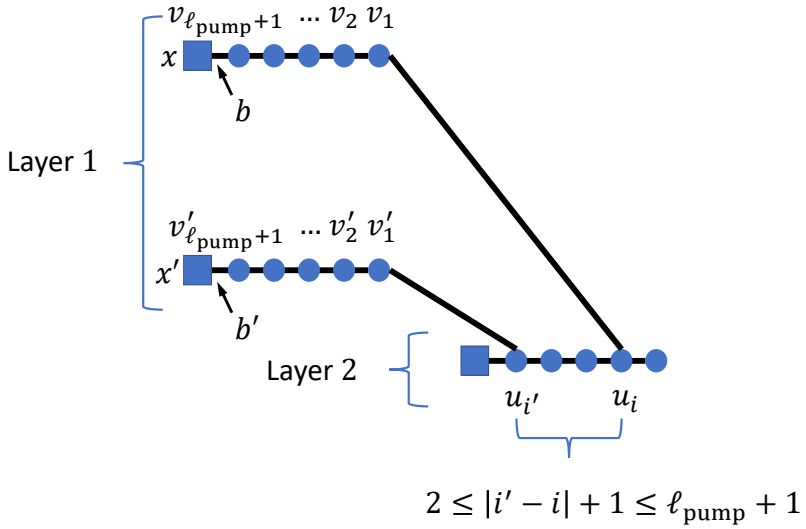

Figure 10: An illustration for the proof of Lemma 16.

In the subsequent discussion, the length of a path refers to the number of edges in a path. In particular, the length of a k-vertex path is $k-1$.

The following lemma is proved by iteratively applying Lemma 16 via intermediate vertex configurations $\tilde{x} \in \mathcal{V}^*$.

Lemma 17. *For any $\mathtt{a} \in x \in \mathcal{V}^*$ and $\mathtt{a}' \in x' \in \mathcal{V}^*$, we have $(x, \mathtt{b}) \overset{k}{\leftrightarrow} (x', \mathtt{b}')$ for some $\mathtt{b} \in x \setminus \{\mathtt{a}\}$ and $\mathtt{b}' \in x' \setminus \{\mathtt{a}'\}$ for all $k \ge 6\ell_{\text{pump}} + 10$.*

Proof. By applying Lemma 16 for three times, we infer that Lemma 16 also works for any path length $k - 1 = (k_1 - 1) + (k_2 - 1) + (k_3 - 1)$, where $k_i - 1 = 2\ell_{\text{pump}} + 3 + t_i$ for $0 \le t_i \le \ell_{\text{pump}} - 1$.

Specifically, we first apply Lemma 16 to find a path realizing $(x, \mathtt{b}) \overset{k_1}{\leftrightarrow} \tilde{x}$ for some $\tilde{x} \in \mathcal{V}^*$, and for some $\mathtt{b} \in x \setminus \{\mathtt{a}\}$. Let $\tilde{\mathtt{a}} \in \tilde{x}$ be the real half-edge label used in the last vertex of the path. We extend the length of this path by $k_2 - 1$ by attaching to it the path realizing $(\tilde{x}, \tilde{\mathtt{b}}) \overset{k_2}{\leftrightarrow} \tilde{x}$, for some $\tilde{\mathtt{b}} \in \tilde{x} \setminus \{\tilde{\mathtt{a}}\}$. Finally, let $\tilde{\mathtt{c}} \in \tilde{x}$ be the real half-edge label used in the last vertex of the current path. We extend the length of the current path by k_3 by attaching to it the path realizing $(\tilde{x}, \tilde{\mathtt{d}}) \overset{k_3}{\leftrightarrow} (x', \mathtt{b}')$, for some $\tilde{\mathtt{d}} \in \tilde{x} \setminus \{\tilde{\mathtt{x}}\}$, and for some $\mathtt{b}' \in x' \setminus \{\mathtt{a}'\}$. The resulting path realizes $(x, \mathtt{b}) \overset{k}{\leftrightarrow} (x', \mathtt{b}')$ for some $\mathtt{b} \in x \setminus \{\mathtt{a}\}$ and $\mathtt{b}' \in x' \setminus \{\mathtt{a}'\}$.

Therefore, Lemma 16 also works with path length of the form $k - 1 = 6\ell_{\text{pump}} + 9 + t$, for any $t \in [0, 3\ell_{\text{pump}} - 3]$.

Any $k - 1 \ge 6\ell_{\text{pump}} + 9$ can be written as $k - 1 = b(2\ell_{\text{pump}} + 3) + (6\ell_{\text{pump}} + 9 + t)$, for some $t \in [0, 3\ell_{\text{pump}} - 3]$ and some integer $b \ge 0$. Therefore, similar to the above, we can find a path

showing $(x, \mathtt{b}) \overset{k}{\leftrightarrow} (x', \mathtt{b}')$ by first applying Lemma 16 with path length $2\ell_{\text{pump}} + 3$ for b times, and then applying the above variant of Lemma 16 to extend the path length by $6\ell_{\text{pump}} + 9 + t$. □

We show that Lemmas 15 and 17 imply that $\mathcal{V}'$ is ℓ-full for some ℓ.

Lemma 18 ($\mathcal{V}'$ is ℓ-full)**.** *The set $\mathcal{V}'$ is ℓ-full for $\ell = 10\ell_{\text{pump}} + 10$.*

Proof. We show that for any target path length $k - 1 \geq 10\ell_{\text{pump}} + 9$, and for any $\mathtt{a} \in x \in \mathcal{V}^*$ and $\mathtt{a}' \in x' \in \mathcal{V}^*$, we have $(x, \mathtt{a}) \overset{k}{\leftrightarrow} (x', \mathtt{a}')$.

By Lemma 15, there exists a vertex configuration $\tilde{x} \in \mathcal{V}^*$ so that we can find a path P realizing $(x, \mathtt{a}) \overset{\ell_1}{\leftrightarrow} \tilde{x}$ for some path length $1 \leq (\ell_1 - 1) \leq 2\ell_{\text{pump}}$. Similarly, there exists a vertex configuration $\tilde{x}' \in \mathcal{V}^*$ so that we can find a path P' realizing $(x', \mathtt{a}') \overset{\ell_2}{\leftrightarrow} \tilde{x}'$ for some path length $1 \leq (\ell_2 - 1) \leq 2\ell_{\text{pump}}$.

Let $\tilde{\mathtt{a}}$ be the real half-edge label for $\tilde{x}$ in P, and let $\tilde{\mathtt{a}}'$ be the real half-edge label for $\tilde{x}'$ in P'. We apply Lemma 17 to find a path $\tilde{P}$ realizing $(x, \mathtt{b}) \overset{\tilde{\ell}}{\leftrightarrow} (x', \mathtt{b}')$ for some $\mathtt{b} \in \tilde{x} \setminus \{\tilde{\mathtt{a}}\}$ and $\mathtt{b}' \in \tilde{x}' \setminus \{\tilde{\mathtt{a}}'\}$ with path length

$$\tilde{\ell} - 1 = (k - 1) - (\ell_1 - 1) - (\ell_2 - 1) \geq 6\ell_{\text{pump}} + 9.$$

The path formed by concatenating P_1, $\tilde{P}$, and P_2 shows that $(x, \mathtt{a}) \overset{k}{\leftrightarrow} (x', \mathtt{a}')$. □

A Feasible Labeling Function f exists We show that a feasible labeling function f exists given that Π can be solved in $\mathsf{LOCAL}(n^{o(1)})$ rounds. We will construct a labeling function f in such a way each equivalence class in $\mathsf{Class}_1(W^*)$ contains only rooted trees that admit legal labeling. That is, for each $c \in \mathsf{Class}_1(W^*)$, the mapping h associated with c satisfies $h(X) = \mathsf{YES}$ for some X.

We will consider a series of modifications in the construction

$$W \to X_f(W) \to Y_f(W) \to Z_f(W)$$

that do not alter the sets of equivalence classes in these sets, conditioning on the assumption that all trees that we have processed admit correct labelings. That is, whether f is feasible is invariant of the modifications.

Applying the Pumping Lemma Let $w > 0$ be some target length for the pumping lemma. In the construction of $Y_f(W)$, when we process a bipolar tree

$$H = (T_1^l, T_2^l, \ldots, T_{\ell_{\text{pump}}}^l, T^m, T_1^r, T_2^r, \ldots, T_{\ell_{\text{pump}}}^r),$$

we apply Lemma 7 to the two subtrees $H^l = (T_1^l, T_2^l, \ldots, T_{\ell_{\text{pump}}}^l)$ and $H^r = (T_1^r, T_2^r, \ldots, T_{\ell_{\text{pump}}}^r)$ to obtain two new bipolar trees H_+^l and H_+^r. The s-t path in the new trees H_+^l and H_+^r contains $w + x$ vertices, for some $0 \leq x < \ell_{\text{pump}}$. The equivalence classes do not change, that is, $\mathsf{Class}_2(H^l) = \mathsf{Class}_2(H_+^l)$ and $\mathsf{Class}_2(H^r) = \mathsf{Class}_2(H_+^r)$.

We replace H^l by H_+^l and replace H^r by H_+^r in the bipolar tree H. Recall that the outcome of applying the labeling function f to the root r of the rooted tree $T_{\ell_{\text{pump}}+1}$ depends only on

$$\mathsf{Class}_2(H^l), \mathsf{Class}_1(T_1^m), \mathsf{Class}_1(T_2^m), \ldots, \mathsf{Class}_1(T_{\Delta-2}^m), \mathsf{Class}_2(H^r),$$

so applying the pumping lemma to H^l and H^r during the construction of $Y_f(W)$ does not alter $\mathsf{Class}_1(T_*^m)$ and $\mathsf{Class}_2(H^*)$ for the resulting bipolar tree H^* and its middle rooted tree T_*^m, by Lemma 8.

Reusing Previous Trees During the construction of the fixed point W^*, we remember all bipolar trees to which we have applied the feasible function f.

During the construction of $Y_f(W)$. Suppose that we are about to process a bipolar tree H, and there is already some other bipolar tree $\tilde{H}$ to which we have applied f before so that $\mathsf{Class}_2(H^l) = \mathsf{Class}_2(\tilde{H}^l)$, $\mathsf{Class}_1(T_i^m) = \mathsf{Class}_1(\tilde{T}_i^m)$ for each $1 \le i \le \Delta - 2$, and $\mathsf{Class}_2(H^r) = \mathsf{Class}_2(\tilde{H}^r)$. Then we replace H by $\tilde{H}$, and then we process $\tilde{H}$ instead.

By Lemma 8, this modification does not alter $\mathsf{Class}_1(T_*^m)$ and $\mathsf{Class}_2(H^*)$ for the resulting bipolar tree H^*.

Not Cutting Bipolar Trees We consider the following different construction of $Z_f(W)$ from $Y_f(W)$. For each $H^* \in Y_f(W)$, we simply add two copies of H^* to $Z_f(W)$, one of them has $r = s$ and the other one has $r = t$. That is, we do not cut the bipolar tree H^*, as in the original construction of $Z_f(W)$.

In general, this modification might alter $\mathsf{Class}_1(Z_f(W))$. However, it is not hard to see that if all trees in $Y_f(W)$ already admit correct labelings, then $\mathsf{Class}_1(Z_f(W))$ does not alter after the modification.

Specifically, let T be a rooted tree in $Z_f(W)$ with the above modification. Then T is identical to a bipolar tree $H^* = (T_1, T_2, \ldots, T_k) \in Y_f(W)$, where there exists some $1 < i < k$ such that the root r_i of T_i has its half-edge labels fixed by f. Suppose that the root of T is the root r_1 of T_1. If we do not have the above modification, then the rooted tree T' added to $Z_f(W)$ corresponding to T is $(T_1, T_2, \ldots, T_i)$, where the root of T' is also r_1.

Given the assumption that all bipolar trees in $Y_f(W)$ admit correct labelings, it is not hard to see that $\mathsf{Class}_1(T') = \mathsf{Class}_1(T)$. For any correct labeling $\mathcal{L}'$ of T', we can extend the labeling to a correct labeling $\mathcal{L}$ of T by labeling the remaining vertices according to any arbitrary correct labeling of H^*. This is possible because the labeling of r_i has been fixed. For any correct labeling $\mathcal{L}$ of T, we can obtain a correct labeling $\mathcal{L}'$ of T' by restricting $\mathcal{L}$ to T'.

Simulating a LOCAL Algorithm We will show that there is a labeling function f that makes all the rooted trees in W^* to admit correct solutions, where we apply the above three modifications in the construction of W^*. In view of the above discussion, such a function f is feasible.

The construction of W^* involves only a finite number k of iterations of Z_f applied to some previously constructed rooted trees. If we view the target length w of the pumping lemma as a variable, then the size of a tree in W^* can be upper bounded by $O(w^k)$.

Suppose that we are given an arbitrary $n^{o(1)}$-round LOCAL algorithm $\mathcal{A}$ that solves Π. It is known [35, 32] that randomness does not help for LCL problems on bounded-degree trees with round complexity $\Omega(\log n)$, so we assume that $\mathcal{A}$ is deterministic.

The runtime of $\mathcal{A}$ on a tree of size $O(w^k)$ can be made to be at most $t = w/10$ if w is chosen to be sufficiently large.

We pick the labeling function f as follows. Whenever we encounter a new bipolar tree H to which we need to apply f, we simulate $\mathcal{A}$ locally at the root r^m of the middle rooted tree T^m in H. Here we assume that the pumping lemma was applied before simulating $\mathcal{A}$. Moreover, when we do the simulation, we assume that the number of vertices is $n = O(w^k)$.

To make the simulation possible, we just locally generate arbitrary distinct identifiers in the radius-t neighborhood S of r^m. This is possible because $t = w/10$ is so small that for any vertex v in any tree T constructed by recursively applying Z_f for at most k iterations, the radius-$(t+1)$ neighborhood of v intersects at most one such S-set. Therefore, it is possible to complete the

identifier assignment to the rest of the vertices in T so that for any edge $\{u, v\}$, the set of identifiers seen by u and v are distinct in an execution of a t-round algorithm.

Thus, by the correctness of $\mathcal{A}$, the labeling of all edges $\{u, v\}$ are configurations in $\mathcal{E}$ and the labeling of all vertices v are configurations in $\mathcal{V}$. Our choice of f implies that all trees in W^* admit correct labeling, and so f is feasible. Combined with Lemma 18, we have proved that $\mathsf{LOCAL}(n^{o(1)}) \Rightarrow \ell$-full.

7 Open Problems

Here we collect some open problems we find interesting.

1. What is the relation of $\mathsf{ffiid}, \mathsf{MEASURE}, \mathsf{fiid}$? In particular is perfect matching in the class ffiid or $\mathsf{MEASURE}$?

2. Are there LCL problems whose uniform complexity is nontrivial and slower than $\operatorname{poly}\log 1/\varepsilon$?

3. Suppose that Π_G is a homomorphism LCL, is it true that $\Pi_G \in \mathsf{BOREL}$ if and only if $\Pi_G \in O(\log^*(n))$?

4. Are there finite graphs of chromatic number bigger than $2\Delta - 2$ such that the corresponding homomorphism LCL is not in BOREL? See also Remark 4.

Acknowledgments

We would like to thank Anton Bernshteyn, Endre Csóka, Mohsen Ghaffari, Jan Hladký, Steve Jackson, Alexander Kechris, Edward Krohne, Oleg Pikhurko, Brandon Seward, Jukka Suomela, and Yufan Zheng for insightful discussions. Yi-Jun Chang would like to thank Yufan Zheng for a discussion about homomorphism LCL problems, in particular for suggesting potentially hard instances.

References

[1] Omer Angel, Itai Benjamini, Ori Gurel-Gurevich, Tom Meyerovitch, and Ron Peled. Stationary map coloring. *Ann. Inst. Henri Poincaré Probab. Stat.*, 48(2):327–342, 2012.

[2] Ágnes Backhausz, Balázs Gerencsér, and Viktor Harangi. Entropy inequalities for factors of IID. *Groups Geom. Dyn.*, 13(2):389–414, 2019. doi:10.4171/GGD/492.

[3] Ágnes Backhausz and Balázs Szegedy. On large girth regular graphs and random processes on trees. *Random Structures Algorithms*, 53(3):389–416, 2018.

[4] Ágnes Backhausz, Balázs Szegedy, and Bálint Virág. Ramanujan graphings and correlation decay in local algorithms. *Random Structures Algorithms*, 47(3):424–435, 2015. doi:10.1002/rsa.20562.

[5] Ágnes Backhausz and Bálint Virag. Spectral measures of factor of i.i.d. processes on vertex-transitive graphs. *Ann. Inst. Henri Poincaré Probab. Stat.*, 53(4):2260–2278, 2017.

[6] Ágnes Backhausz, Balázs Gerencsér, Viktor Harangi, and Máté Vizer. Correlation bounds for distant parts of factor of iid processes. *Combin. Probab. Comput.*, 27(1):1–20, 2018.

[7] K. Ball. Poisson thinning by monotone factors. *Electron. Comm. Probab.*, 10:60–69, 2005.

[8] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2020/13095`, `doi:10.4230/LIPIcs.DISC.2020.17`.

[9] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *Proceedings of the IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 481–497, 2019. `doi:10.1109/FOCS.2019.00037`.

[10] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Improved distributed lower bounds for mis and bounded (out-)degree dominating sets in trees. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC)*, 2021.

[11] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 365–376, 2020. `doi:10.1109/FOCS46700.2020.00042`.

[12] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. Locally checkable problems in rooted trees. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021.

[13] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the local model. *Distributed Computing*, pages 1–23, 2020.

[14] Alkida Balliu, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1307–1318, 2018.

[15] Alkida Balliu, Juho Hirvonen, Dennis Olivetti, and Jukka Suomela. Hardness of minimal symmetry breaking in distributed computing. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 369–378, 2019.

[16] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM*, 63(3):20, 2016.

[17] Ferenc Bencs, Aranka Hrušková, and László Márton Tóth. Factor of iid schreier decoration of transitive graphs. *arXiv:2101.12577*, 2021.

[18] Itai Benjamini and Oded Schramm. Recurrence of distributional limits of finite planar graphs. In *Selected Works of Oded Schramm*, pages 533–545. Springer, 2011.

[19] Anton Bernshteyn. Work in progress.

[20] Anton Bernshteyn. Distributed algorithms, the Lovász Local Lemma, and descriptive combinatorics. *arXiv preprint arXiv:2004.04905*, 2020.

[21] Anton Bernshteyn. A fast distributed algorithm for $(\Delta+1)$-edge-coloring. *preprint*, 2020.

[22] Anton Bernshteyn. Probabilistic constructions in continuous combinatorics and a bridge to distributed algorithms., 2021.

[23] Béla Bollobás. The independence ratio of regular graphs. *Proc. Amer. Math. Soc.*, 83(2):433–436, 1981. doi:10.2307/2043545.

[24] Lewis Bowen. The ergodic theory of free group actions: entropy and the f -invariant. *Groups Geom. Dyn.*, 4(3):419–432, 2010.

[25] Lewis Bowen. A measure-conjugacy invariant for free group actions. *Ann. of Math.*, 171(2):1387–1400, 2010.

[26] Sebastian Brandt. An automatic speedup theorem for distributed problems. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 379–388, 2019. doi:10.1145/3293611.3331611.

[27] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, pages 479–488, 2016.

[28] Sebastian Brandt, Christoph Grunau, and Václav Rozhoň. The landscape of distributed complexities on trees, 2021.

[29] Sebastian Brandt, Christoph Grunau, and Václav Rozhoň. Randomized local computation complexity of the Lovasz local lemma. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021.

[30] Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R.J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. LCL problems on grids. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, page 101–110, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3087801.3087833.

[31] Sebastian Brandt and Dennis Olivetti. Truly tight-in-Δ bounds for bipartite maximal matching and variants. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, page 69–78, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3382734.3405745.

[32] Yi-Jun Chang. The Complexity Landscape of Distributed Locally Checkable Problems on Trees. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing (DISC 2020)*, volume 179 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2020/13096, doi:10.4230/LIPIcs.DISC.2020.18.

[33] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. Distributed edge coloring and a special case of the constructive Lovász local lemma. *ACM Trans. Algorithms*, 16(1):8:1–8:51, 2020. doi:10.1145/3365004.

[34] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the local model. *SIAM Journal on Computing*, 48(1):122–143, 2019.

[35] Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the local model. *SIAM Journal on Computing*, 48(1):33–69, 2019.

[36] Yi-Jun Chang, Jan Studený, and Jukka Suomela. Distributed graph problems through an automata-theoretic lens. *arXiv:2002.07659*, 2020.

[37] Sourav Chatterjee, Ron Peled, Yuval Peres, and Dan Romik. Gravitational allocation to poisson points. *Ann. of Math.*, 172(1):617–671, 2010.

[38] Clinton T. Conley, Steve Jackson, Andrew S. Marks, Brandon Seward, and Robin Tucker-Drob. Borel asymptotic dimension and hyperfinite equivalence relations. *arXiv:2009.06721*, 2020.

[39] Clinton T. Conley, Andrew S. Marks, and Robin D. Tucker-Drob. Brooks' theorem for measurable colorings. *Forum Math. Sigma*, e16(4):23pp, 2016.

[40] Clinton T. Conley, Andrew S. Marks, and Spencer T. Unger. Measurable realizations of abstract systems of congruences. In *Forum of Mathematics, Sigma*, volume 8. Cambridge University Press, 2020.

[41] Clinton T. Conley and Benjamin D. Miller. A bound on measurable chromatic numbers of locally finite Borel graphs. *Math. Res. Lett.*, 23(6):1633–1644, 2016. doi:10.4310/MRL.2016.v23.n6.a3.

[42] Clinton T. Conley and Benjamin D. Miller. Measure reducibility of countable Borel equivalence relations. *Ann. of Math. (2)*, 185(2):347–402, 2017. doi:10.4007/annals.2017.185.2.1.

[43] Endre Csóka, Balázs Gerencsér, Viktor Harangi, and Bálint Virág. Invariant gaussian processes and independent sets on regular graphs of large girth. *Random Structures & Algorithms*, 47(2):284–303, 2015.

[44] Endre Csóka, Łukasz Grabowski, András Máthé, Oleg Pikhurko, and Konstantinos Tyros. Borel version of the Local Lemma. *arXiv:1605.04877*, 2017.

[45] Randall Dougherty and Matthew Foreman. Banach-Tarski paradox using pieces with the property of Baire. *Proc. Nat. Acad. Sci. U.S.A.*, 89(22):10726–10728, 1992. doi:10.1073/pnas.89.22.10726.

[46] Randall Dougherty, Steve Jackson, and Alexander S. Kechris. The structure of hyperfinite Borel equivalence relations. *Trans. Amer. Math. Soc.*, 341(1):193–225, 1994. doi:10.2307/2154620.

[47] Gábor Elek and Gábor Lippner. Borel oracles. An analytical approach to constant-time algorithms. *Proc. Amer. Math. Soc.*, 138(8):2939–2947, 2010. doi:10.1090/S0002-9939-10-10291-3.

[48] Jens E. Fenstad and Dag Normann. On absolutely measurable sets. *Fund. Math.*, 81(2):91–98, 1973/74. doi:10.4064/fm-81-2-91-98.

[49] Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for Lovàsz local lemma, and the complexity hierarchy. In *Proceedings of the 31st Symposium on Distributed Computing (DISC)*, pages 18:1–18:16, 2017.

[50] Klaus-Tycho Foerster, Janne H. Korhonen, Ami Paz, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Work in progress.

[51] Damien Gaboriau. Coût des relations d'équivalence et des groupes. *Invent. Math.*, 139(1):41–98, 2000. doi:10.1007/s002229900019.

[52] Damien Gaboriau and Russell Lyons. A measurable-group-theoretic solution to von neumann's problem. *Invent. Math.*, 177(3):533–540, 2009.

[53] David Gamarnik and Madhu Sudan. Limits of local algorithms over sparse random graphs. *Proceedings of the 5-th Innovations in Theoretical Computer Science conference, ACM Special Interest Group on Algorithms and Computation Theory*, 2014.

[54] Su Gao and Steve Jackson. Countable abelian group actions and hyperfinite equivalence relations. *Inventiones Math.*, 201(1):309–383, 2015.

[55] Su Gao, Steve Jackson, Edward Krohne, and Brandon Seward. Forcing constructions and countable borel equivalence relations. *arXiv:1503.07822*, 2015.

[56] Su Gao, Steve Jackson, Edward Krohne, and Brandon Seward. Continuous combinatorics of abelian group actions., 2018.

[57] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 270–277, 2016.

[58] Mika Göös, Juho Hirvonen, and Jukka Suomela. Linear-in-Δ lower bounds in the LOCAL model. In *Proc. 33rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 86–95, 2014.

[59] Łukasz Grabowski, András Máthé, and Oleg Pikhurko. Measurable circle squaring. *Ann. of Math. (2)*, 185(2):671–710, 2017. doi:10.4007/annals.2017.185.2.6.

[60] Jan Grebík and Oleg Pikhurko. Measurable versions of vizing's theorem. *Advances in Mathematics*, 374:107378, 2020.

[61] Jan Grebík and Václav Rozhoň. Classification of local problems on paths from the perspective of descriptive combinatorics. *arXiv preprint arXiv:2103.14112*, 2021. arXiv:2103.14112.

[62] Jan Grebík and Václav Rozhoň. Local problems on grids from the perspective of distributed algorithms, finitary factors, and descriptive combinatorics. *arXiv preprint arXiv:2103.08394*, 2021. arXiv:2103.08394.

[63] Ori Gurel-Gurevich and Ron Peled. Poisson thickening. *Israel J. Math.*, 196(1):215–234, 2013.

[64] Viktor. Harangi and Bálint Virág. Independence ratio and random eigenvectors in transitive graphs. *Ann. Probab.*, 43(5):2810–2840, 2015.

[65] Hamed Hatami, László Lovász, and Balázs Szegedy. Limits of locally-globally convergent graph sequences. *Geom. Funct. Anal.*, 24(1):269–296, 2014. doi:10.1007/s00039-014-0258-7.

[66] Greg Hjorth and Alexander S. Kechris. Borel equivalence relations and classifications of countable models. *Annals of pure and applied logic*, 82(3):221–272, 1996.

[67] Alexander E. Holroyd. Geometric properties of poisson matchings. *Probab. Theory Related Fields*, 150(3–4):511–527, 2011.

[68] Alexander E. Holroyd. One-dependent coloring by finitary factors. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 53(2):753 – 765, 2017. doi:10.1214/15-AIHP735.

[69] Alexander E. Holroyd, Russel Lyons, and Terry Soo. Poisson splitting by factors. *Ann. Probab.*, 39(5):1938–1982, 2011.

[70] Alexander E. Holroyd, Robin Pemantle, Y. Peres, and Oded Schramm. Poisson matching. *Ann. Inst. Henri Poincaré Probab. Stat.*, 45(1):266–287, 2009.

[71] Alexander E. Holroyd and Yuval Peres. Trees and matchings from point processes. *Electron. Comm. Probab.*, 8:17–27, 2003.

[72] Alexander E. Holroyd, Oded Schramm, and David B. Wilson. Finitary coloring. *Ann. Probab.*, 45(5):2867–2898, 2017.

[73] Carlos Hoppen and Nicholas Wormald. Local algorithms, regular graphs of large girth, and random regular graphs. *Combinatorica*, 38(3):619–664, 2018.

[74] Steve Jackson, Alexander S. Kechris, and Alain Louveau. Countable Borel equivalence relations. *J. Math. Logic*, 2(01):1–80, 2002.

[75] Alexander S. Kechris. *Classical descriptive set theory*, volume 156 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. doi:10.1007/978-1-4612-4190-4.

[76] Alexander S. Kechris and Andrew S. Marks. Descriptive graph combinatorics. *http://www.math.caltech.edu/~kechris/papers/combinatorics20book.pdf*, 2020.

[77] Alexander S. Kechris, Sławomir Solecki, and Stevo Todorčević. Borel chromatic numbers. *Adv. Math.*, 141(1):1–44, 1999.

[78] Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. Toward more localized local algorithms: removing assumptions concerning global knowledge. *Distributed Computing*, 26(5-6):289–308, Sep 2012. URL: http://dx.doi.org/10.1007/s00446-012-0174-8, doi:10.1007/s00446-012-0174-8.

[79] Gábor Kun. Expanders have a spanning lipschitz subgraph with large girth. *preprint*, 2013.

[80] Miklós Laczkovich. Equidecomposability and discrepancy; a solution of Tarski's circle-squaring problem. *J. Reine Angew. Math.*, 404:77–117, 1990. doi:10.1515/crll.1990.404.77.

[81] Nati Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[82] Russell Lyons. Factors of iid on trees. *Combin. Probab. Comput.*, 26(2):285–300, 2017.

[83] Russell Lyons and Fedor Nazarov. Perfect matchings as iid factors on non-amenable groups. *European Journal of Combinatorics*, 32(7):1115–1125, 2011.

[84] Andrew Marks and Spencer Unger. Baire measurable paradoxical decompositions via matchings. *Advances in Mathematics*, 289:397–410, 2016.

[85] Andrew S. Marks. A short proof that an acyclic n-regular Borel graph may have borel chromatic number $n + 1$.

[86] Andrew S. Marks. A determinacy approach to Borel combinatorics. *J. Amer. Math. Soc.*, 29(2):579–600, 2016.

[87] Andrew S. Marks and Spencer T. Unger. Borel circle squaring. *Ann. of Math. (2)*, 186(2):581–605, 2017. doi:10.4007/annals.2017.186.2.4.

[88] Donald A. Martin. Borel determinacy. *Ann. of Math. (2)*, 102(2):363–371, 1975. doi:10.2307/1971035.

[89] Colin McDiarmid. Concentration for independent permutations. *Combinatorics Probability and Computing*, 11(2):163–178, 2002.

[90] Brendan D. McKay, Nicholas C. Wormald, and Beata Wysocka. Short cycles in random regular graphs. *Electron. J. Combin.*, 11(1):Research Paper 66, 12, 2004. URL: http://www.combinatorics.org/Volume_11/Abstracts/v11i1r66.html.

[91] Péter Mester. A factor of i.i.d with uniform marginals and infinite clusters spanned by equal labels. *preprint*, 2011.

[92] Gary L. Miller and John H. Reif. Parallel tree contraction–Part I: fundamentals. *Advances in Computing Research*, 5:47–72, 1989.

[93] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[94] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 327–336, 2008. doi:10.1109/FOCS.2008.81.

[95] Oleg Pikhurko. Borel combinatorics of locally finite graphs. *arXiv preprint arXiv:2009.09113*, 2021. arXiv:2009.09113.

[96] Mustazee Rahman. Factor of iid percolation on trees. *SIAM J. Discrete Math.*, 30(4):2217–2242, 2016.

[97] Mustazee Rahman and Bálint Virág. Local algorithms for independent sets are half-optimal. *Ann. Probab.*, 45(3):1543–1577, 2017.

[98] Christian Schmidt, Nils-Eric Guenther, and Lenka Zdeborová. Circular coloring of random graphs: statistical physics investigation. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(8):083303, 2016.

[99] Brandon Seward. Personal communication.

[100] Yaroslav Shitov. Counterexamples to Hedetniemi's conjecture. *Ann. of Math. (2)*, 190(2):663–667, 2019. doi:10.4007/annals.2019.190.2.6.

[101] Robert M. Solovay. *unpublished.*

[102] Yinon Spinka. Finitely dependent processes are finitary. *Ann. Probab.*, 48(4):2088–2117, 2020.

[103] Claude Tardif and Xuding Zhu. A note on Hedetniemi's conjecture, Stahl's conjecture and the Poljak-Rödl function. *arXiv preprint arXiv:1906.03748*, 2019.

[104] Ádám Timár. Tree and grid factors for general point processes. *Electron. Comm. Probab.*, 9(53–59), 2004.

[105] Ádám Timár. Invariant colorings of random planar maps. *Ergodic Theory Dynam. Systems*, 31(2):549–562, 2011.

[106] Stevo Todorčević and Zoltán Vidnyánszky. A complexity problem for Borel graphs. *Invent. Math.*, 226:225–249, 2021.

[107] Felix Weilacher. Borel vizing's theorem for 2-ended groups. *arXiv:2101.12740*, 2021.

[108] Nicholas C. Wormald. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.

[109] Xuding Zhu. Note on Hedetniemi's conjecture and the Poljak-Rödl function. *2019-20 MATRIX Annals*, pages 499–511, 2021.

A Missing Proof of Lemma 2

Proof. We define an acyclic Δ regular graph $\mathcal{T}$ on some standard probability space (X, μ) and then show that any measurable solution of Π on $\mathcal{T}$ implies that $\Pi \in \mathsf{fiid}$.

Suppose that there is a fixed distinguished vertex $*$ in T_Δ, the root. Let $\mathrm{Aut}^*(T_\Delta)$ be the subgroup that fixes the root. Then it is easy to see that $\mathrm{Aut}^*(T_\Delta)$ is a compact group. It is a basic fact that $Y = [0,1]^{T_\Delta}$ with the product Borel structure is a standard Borel space and the product Lebesgue measure λ is a Borel probability measure on Y. Consider the shift action $\mathrm{Aut}^*(T_\Delta) \curvearrowright Y$ defined as

$$\alpha \cdot x(v) = x(\alpha^{-1}(v)),$$

where $v \in T_\Delta$ and $\alpha \in \mathrm{Aut}^*(T_\Delta)$. Since $\mathrm{Aut}^*(T_\Delta)$ is compact, we have that the quotient space

$$X := [0,1]^{T_\Delta} / \mathrm{Aut}^*(T_\Delta)$$

is a standard Borel space, see [75]. Moreover, since the shift action preserves λ, we have that

$$\mu := \lambda / \mathrm{Aut}^*(T_\Delta)$$

is a Borel probability measure on X.

We define $\mathcal{T}$ on (X, μ) as follows. Let $[x], [y] \in X$, where $x, y \in Y$ and $[-]$ denotes the $\mathrm{Aut}^*(T_\Delta)$-equivalence class. We put $([x], [y]) \in \mathcal{T}$ if and only if there is a neighbor v of the root in x such that y is isomorphic to x with v as the root. Note that this definition is independent of the choice of representatives of $[x]$ and $[y]$. Moreover, $\mathcal{T}$ is acyclic and Δ-regular with probability 1. A standard argument shows that every measurable solution to an LCL Π on $\mathcal{T}$ yields a fiid solution by composition with the quotient map. □

B Missing Proof of Theorem 10

Proof. It remains to verify that the LLL algorithms of Fischer and Ghaffari [49, Section 3.1.1] and Chang et al. [33, Section 5.1] can be turned into an algorithm that does not rely on the knowledge of n. As discussed in [33, Section 5.1], these two algorithms can be combined together to solve the tree-structured LLL problem in $O(\log \log n)$ rounds when the underlying tree has bounded degree.

Before we explain the main ideas in these algorithms, consider the following setup. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two distributed algorithms that do not rely on knowing n and that have a round complexity of $g_1(n)$ and $g_2(n)$, respectively. We now want to compose these two algorithms in the following sense. We are interested in the output of $\mathcal{A}_2$ when the input of $\mathcal{A}_2$ is equal to the output of $\mathcal{A}_1$. Then, we can compute this composition with a distributed algorithm that works without the knowledge of n and which has a round complexity of $O(g_1(n) + g_2(n))$. This statement might seem obvious at first sight, but one needs to be careful. In particular, as the resulting algorithm needs to work without the knowledge of n it cannot compute the value $g_1(n)$. Therefore, it cannot explicitly tell the vertices in which round they should start to run the second algorithm. However, this is not a problem as vertices can start to run the second algorithm as soon as their neighbors are ready. In subsequent rounds, vertices might need to temporarily stall as some neighbors might not have received all the necessary information from all of their neighbors and so on, but this is not a problem. From this discussion, it should be easy to see that the algorithm indeed finishes after $O(g_1(n) + g_2(n))$ rounds. In case the algorithms are randomized the failure probability of the resulting algorithm might increase up to a factor of 2.

This composition result makes it easier to verify that the algorithms of [33, 49] indeed works without the knowledge of n, as we can verify that this is the case for all its subroutines in isolation.

The pre-shattering phase of the LLL algorithm [49, Section 3.1.1] consists of first computing a $\text{poly}(\Delta)$-coloring of T^k — the graph obtained from T by connecting any two vertices of distance at most k in T by an edge — for some $k = O(1)$. It directly follows from the results of [78, 72] — they adapt Linial's coloring algorithm in such a way that it works without the knowledge of n — that the $\text{poly}(\Delta)$-coloring of T^k can be computed without the knowledge of n. Afterwards, an $O(1)$-round routine follows that only uses the computed coloring as its input. Hence, the pre-shattering phase works without the knowledge of n. Once the pre-shattering phase is complete a subset of the vertices "survive" and the post-shattering phase is executed on the graph induced by these vertices. Importantly, with high probability all the connected components of the induced graph have size $O(\log n)$. See [49, Lemma 6].

The post-shattering phase of the LLL algorithm [33, Section 5.1] starts by decomposing each connected component with the following variant of the rake-and-compress process, which consists of a repeated application of the following two operations.

Rake: Remove all leaves and isolated vertices.

Compress: Remove all vertices that belong to some path P such that (i) all vertices in P have degree at most 2 and (ii) the number of vertices in P is at least a fixed constant ℓ.

Alternating these two operations on a tree with N vertices decomposes the whole tree in $O(\log N)$ steps. In our case $N = O(\log n)$ with high probability and therefore the procedure finishes after $O(\log \log n)$ rounds with high probability. Importantly, the rake-and-compress algorithm works without the knowledge of n. At the end of the rake-and-compress procedure, the only important information each vertex needs to know is in which iteration it got removed.

After this information is computed, the algorithm of [33] computes in $O(\log^* n)$ rounds a certain coloring variant on a certain subgraph of the input tree. This subgraph can be locally constructed

given the rake-and-compress decomposition. Again, it is a simple corollary of the results of [78, 72] that this coloring variant can be computed without the knowledge of n in $O(\log^* n)$ rounds.

The coloring variant together with the rake-and-compress decomposition is then used to compute a so-called $(2, O(\log N))$ network decomposition of the graph T^k for some $k = O(1)$ in $O(\log N)$ rounds [33, Section 6.1]. We do not explain how this network decomposition is computed in detail, but the idea is to first compute the local output of all the vertices that got removed in the very last rake-and-compress iteration in $O(1)$ rounds. Directly afterwards, the local output of all the vertices that got removed one iteration before gets computed in additional $O(1)$ rounds and so on. Hence the local information of all vertices is computed within $O(\log N) \cdot O(1) = O(\log N)$ rounds. From this description, it is clear that this procedure works without the knowledge of n.

Finally, once the network decomposition of T^k is computed, it directly follows from the algorithm description of [33] that the final computation can be performed by a distributed algorithm without the knowledge of n in $O(\log N)$ rounds.

Hence we have shown that the $O(\log \log n)$-round LLL algorithm [33, 49] works without the knowledge of n. □

C Missing Proof of Theorem 14

Proof of Theorem 14. Let x be a vertex of degree strictly bigger than 2. A *star* $S(x)$ around x consists of x and vertices of degree 2 that are connected with x by a path with all inner vertices of degree 2. It is clear that $S(x)$ is connected in $\mathcal{G}$.

Recall that if F is a function we define the iterated preimage of a vertex x as $F^{\leftarrow}(x) = \bigcup_{n \in \mathbb{N}} F^{-n}(x)$. We first specify a canonical one ended orientation on $S(x)$, or, equivalently, a function F with finite iterated preimages. (In what follows we interchange freely the notion of one ended orientation and function with finite iterated preimages.) That is to say, if $S(x)$ is infinite we orient things towards infinity in a one ended fashion (that is always possible), otherwise we fix an orientation towards any of the boundary points, i.e., there is exactly one point that is directed outside of $S(x)$. We refer to this orientation as *the canonical orientation*.

The inductive construction produces an orientation of some vertices together with doubly infinite lines. The orientation points either to infinity (is one-ended) or towards these doubly infinite lines, this takes $(\aleph_0 + 1)$-many steps.

For a graph $\mathcal{G}'$ we define a graph $\mathcal{H}'$ on the vertices of degree strictly bigger than 2, where $(x, y) \in \mathcal{H}'$ if there is a path from x to y in $\mathcal{G}'$ that has at most one inner vertex of degree strictly bigger than 2. Since, in our situation, $\mathcal{H}'$ is always Borel and has degree bounded by Δ^2, we can pick a Borel maximal independent set $\mathcal{M}'$ of $\mathcal{H}'$ by [77].

Inductively along $\mathbb{N}$ do the following: Suppose that we are in stage $k \in \mathbb{N}$ and we have a Borel set $\mathcal{O}_k$ and $\mathcal{G}_k := \mathcal{G} \restriction \mathcal{O}_k$ with the property that every vertex has degree at least 2. We start with $\mathcal{G}_0 = \mathcal{G}$ and $\mathcal{O}_0 = X$. Pick a Borel maximal independent set $\mathcal{M}_k$ in $\mathcal{H}_k$ that is defined from $\mathcal{G}_k$. Add to the domain of F every vertex from the star $S_k(x)$ in $\mathcal{G}_k$, where $x \in \mathcal{M}_k$, and set

$$\mathcal{O}_{k+1} = \mathcal{O}_k \setminus \bigcup_{x \in \mathcal{M}_k} S(x).$$

Define F so that it corresponds to the canonical orientation on each $S_k(x)$. By the definition we have that $F(y) \in S_k(x)$ for every $y \in S_k(x)$, possibly up to one point $z \in S_k(x)$ that satisfies $F(z) \in \mathcal{O}_{k+1}$. It is easy to see that every $x \in \mathcal{O}_{k+1}$ has degree at least 2 in $\mathcal{G}_{k+1} = \mathcal{G} \restriction \mathcal{O}_{k+1}$. This follows from the definition of $\mathcal{M}_k$.

Set $\mathcal{O}_\infty = \bigcap_{k\in\mathbb{N}} \mathcal{O}_k$ and write $\mathcal{G}_\infty := \mathcal{G} \restriction \mathcal{O}_\infty$. It follows from the inductive (finitary) construction that every vertex $x \in \mathcal{O}_\infty$ has degree at least 2 in $\mathcal{G}_\infty$. Moreover, the function F satisfies $\mathrm{dom}(F) = X \setminus \mathcal{O}_\infty$ and has finite iterated preimages. This is because, first, for every $x \in \mathrm{dom}(F)$ there is $k \in \mathbb{N}$ such that $x \in \mathcal{O}_k \setminus \mathcal{O}_{k+1}$ and $F^{-1}(x) \subseteq X \setminus \mathcal{O}_{k+1}$. Second, $F^{\leftarrow}(x) \cap \mathcal{O}_k$ is finite by the definition of F on stars. It follows inductively that $F^{\leftarrow}(x) \cap \mathcal{O}_l$ is finite for every $l \leq k$, hence, $F^{\leftarrow}(x)$ is finite, whenever $x \in \mathrm{dom}(F)$. If $x \notin \mathrm{dom}(F)$, then $F^{-1}(x) \subseteq \mathrm{dom}(F)$ is finite and the claim follows.

Let $x \in \mathcal{O}_\infty$ have degree strictly bigger than 2 and write $C_1, \dots, C_\ell$ for the connected components of $\mathcal{G}_\infty \setminus \{x\}$ in the connected component of x in $\mathcal{G}_\infty$. Note that $\ell \leq \Delta$. We claim that at least one of the sets C_i is a one ended line. Suppose not, and write $z_1, \dots, z_\ell$ for the closest splitting points in $C_1, \dots, C_\ell$ and p_i for the paths that connect x with z_i for every $i \leq \ell$. There is $k \in \mathbb{N}$ large enough such that the degree of $x, z_1, \dots, z_\ell$ is the same in $\mathcal{G}_\infty$ as in $\mathcal{G}_k$ and p_i is a path in $\mathcal{G}_k$ whose inner vertices have degree 2. Note that $\mathcal{G}_\infty \subseteq \mathcal{G}_k$ because $\mathcal{O}_\infty \subseteq \mathcal{O}_k$. Since $\mathcal{M}_k$ was maximal in $\mathcal{H}_k$, there is $y \in \mathcal{M}_k$ such that $(x, y) \in \mathcal{H}_k$. This is because $x \notin \mathcal{M}_k$. Similar reasoning implies that that $y \neq z_i$ for any $i \leq \ell$. Let q be the path that connects x and y in $\mathcal{G}_k$. By the choice of k we have that q extends one of the paths p_i. Let y' be the last point on q such that $y' \in \mathcal{O}_\infty$. Since the degree of z_i is the same in $\mathcal{G}_k$ as in $\mathcal{G}_\infty$ we have that $y' \neq z_i$. The degree of y' in $\mathcal{G}_\infty$ is at least 2. Suppose it were 3 in $\mathcal{G}_k$, then $y' = y$ because we must have $(x, y) \in \mathcal{H}_k$. In that case removing $S(y)$ would decrease the degree of z_i in $\mathcal{G}_{k+1}$ and consequently in $\mathcal{G}_\infty$. Therefore y' has degree 2 in $\mathcal{G}_k$. But then y' is not the last vertex on q such that $y' \in \mathcal{O}_\infty$, i.e., the other neighbor of y', that is the same in $\mathcal{G}_k$ and in $\mathcal{G}_\infty$ must be a vertex on q, a contradiction.

Consider now the graph $\mathcal{H}_\infty$, where $(x, y) \in \mathcal{H}_\infty$ if and only if $x, y \in \mathcal{O}_\infty$ have degree strictly bigger than 2 and there is a path from x to y in $\mathcal{G}_\infty$ with all inner points having degree 2 in $\mathcal{G}_\infty$. Consider any Borel $(\Delta + 1)$-coloring of $\mathcal{H}_\infty$ a decomposition of all vertices of degree strictly bigger than 2 in $\mathcal{G}_\infty$ into $\mathcal{H}_\infty$-independent Borel sets $D_0, \dots, D_\Delta$. Define a one ended orientation of stars of the form $S(x)$, where $x \in D_i$, inductively according to $i \leq \Delta$ using the canonical orientation, i.e., in step $i \leq \Delta$ we work with the graph

$$\mathcal{G} \restriction \left(\mathcal{O}_\infty \setminus \bigcup_{j<i} \bigcup_{x\in D_j} S(x) \right).$$

Note that by the previous argument we have that every such star $S(x)$ is infinite and consequently, this defines a valid extension of F, i.e., iterated preimages are still finite. It remains to realize that complement of $\mathrm{dom}(F)$ consists of doubly infinite lines. This finishes the proof. □