Finding List Homomorphisms from Bounded-treewidth Graphs to Reflexive Graphs: a Complete Complexity Characterization

László Egri
Indiana State University, Department of Mathematics & Computer Science, Terre Haute, USA
Laszlo.Egri@indstate.edu

Dániel Marx
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
dmarx@sztaki.mta.hu

Paweł Rzążewski
Faculty of Mathematics and Information Science,
Warsaw University of Technology, Warsaw, Poland
p.rzazewski@mini.pw.edu.pl

Abstract

In the *list homomorphism* problem, the input consists of two graphs G and H, together with a list $L(v) \subseteq V(H)$ for every vertex $v \in V(G)$. The task is to find a homomorphism $\phi : V(G) \to V(H)$ respecting the lists, that is, we have that $\phi(v) \in L(v)$ for every $v \in V(H)$ and if u and v are adjacent in G, then $\phi(u)$ and $\phi(v)$ are adjacent in H. If H is a fixed graph, then the problem is denoted by LHom(H). We consider the *reflexive* version of the problem, where we assume that every vertex in H has a self-loop. If is known that reflexive LHom(H) is polynomial-time solvable if H is an interval graph and it is NP-complete otherwise [Feder and Hell, JCTB 1998].

We explore the complexity of the problem parameterized by the treewidth $\mathrm{tw}(G)$ of the input graph G. If a tree decomposition of G of width $\mathrm{tw}(G)$ is given in the input, then the problem can be solved in time $|V(H)|^{\mathrm{tw}(G)} \cdot n^{O(1)}$ by naive dynamic programming. Our main result completely reveals when and by exactly how much this naive algorithm can be improved. We introduce a simple combinatorial invariant $i^*(H)$, which is based on the existence of certain decompositions and incomparable sets, and show that this number should appear as the base of the exponent in the best possible running time. Specifically, we prove for every non-interval reflexive graph H that

- If a tree decomposition of width $\mathrm{tw}(G)$ is given in the input, then the problem can be solved in time $i^*(H)^{\mathrm{tw}(G)} \cdot n^{O(1)}$.
- Assuming the Strong Exponential-Time Hypothesis (SETH), the problem cannot be solved in time $(i^*(H) - \epsilon)^{\mathrm{tw}(G)} \cdot n^{O(1)}$ for any $\epsilon > 0$.

Thus by matching upper and lower bounds, our result exactly characterizes for every fixed H the complexity of reflexive LHom(H) parameterized by treewidth.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness, Theory of computation → Graph algorithms analysis, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Graph Homomorphism, List Homomorphism, Reflexive Graph, Treewidth

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.27

Funding This work was supported by ERC Starting Grant PARAMTIGHT (No. 280152), ERC Consolidator Grant SYSTEMATICGRAPH (No. 725978) of the European Research Council, and NSF EAGER Grant (No. 1751765).

1 Introduction

It is well known that most NP-hard algorithmic graph problems can be solved significantly more efficiently on graphs of bounded treewidth than on general graphs. A large number of NP-hard problems are known to be fixed-parameter tractable (FPT) parameterized by treewidth, that is, if the input instance contains a tree decomposition of width w of the graph, then the problem can be solved in time $f(w) \cdot n^{O(1)}$ for some computable function f depending only on the width w. In recent years, there have been significant research efforts to understand how complexity depends on treewidth and to determine the best possible function $f(w)$ that can appear in the running time. On the algorithmic side, new algorithms with improved running times were obtained for a number of problems [6, 1, 25, 16]. On the complexity side, conditional lower bounds were given that, in many cases, match the running time of the best known algorithms, thereby giving a tight understanding of the complexity of the problem parameterized by treewidth [21, 20, 6, 22, 5]. These lower bounds are usually based on the Exponential-Time Hypothesis (ETH), which can be informally stated as n-variable 3-Sat cannot be solved in time $2^{o(n)}$, or on the Strong Exponential-Time Hypothesis (SETH), which can be informally stated as n-variable m-clause Cnf-Sat cannot be solved in time $(2-\epsilon)^n \cdot m^{O(1)}$ for any $\epsilon > 0$.

As an exemplary result, let us consider the c-Coloring problem, where the task is to color the vertices of the graph with c colors such that adjacent vertices receive distinct colors. Using standard dynamic programming techniques, c-Coloring can be solved in time $c^{\mathrm{tw}(G)} \cdot n^{O(1)}$ if a tree decomposition of width $\mathrm{tw}(G)$ is given in the input. A result of Lokshtanov et al. [20] showed that this running time is essentially optimal.

▶ **Theorem 1** (Lokshtanov, Marx, and Saurabh [20]). *Let $c \geq 3$ be a fixed integer. Assuming the SETH, the c-Coloring problem on a graph G with n vertices, given with its tree decomposition of width $\mathrm{tw}(G)$, cannot be solved in time poly$(n) \cdot (c-\epsilon)^{\mathrm{tw}(G)}$ for any $\epsilon > 0$.*

Homomorphisms. Given graphs G and H, a *homomorphism* from G to H is a mapping $\phi : V(G) \to V(H)$ such that if uv is an edge of G, then $\phi(u)\phi(v)$ is an edge of H. (In particular, if H has no loops, then this implies $\phi(u) \neq \phi(v)$ whenever u and v are adjacent.) For every fixed graph G, we can define the Hom(H) problem, where, given a graph G, the task is to find a homomorphism from G to H. Now c-Coloring is equivalent to Hom(K_c), where K_c is the clique on c vertices: it is easy to see that G is c-colorable if and only if it has a homomorphism to K_c. Thus the Hom(H) family of problems form a far-reaching generalization of the vertex coloring problem. A classic result of Hell and Nešetřil [17] characterized the complexity of Hom(H): it is polynomial-time solvable if H is bipartite and it is NP-complete for every nonbipartite H (see also [4, 18]).

What can we say about the complexity of Hom(H) parameterized by treewidth? It seems to be a natural goal to try to obtain, for every H, the best possible base c_H of the exponent that can appear in the running time $c_H^{\mathrm{tw}(G)} \cdot n^{O(1)}$. If H is the clique K_c, then we know from Theorem 1 that $c_H = c$, but what can we say about other graphs H? While this is a very natural question, it appears to be very difficult and deep as well: while the hardness of c-Coloring is well understood and can be easily exploited in hardness proofs such as Theorem 1, the hardness of Hom(H) for nonbipartite H comes from a somewhat mysterious combination of combinatorics and algebra [17, 4, 18, 23].

While we are unable at the moment to characterize the exact complexity of Hom(H) parameterized by treewidth, we resolve a related question that is still of interest, but apparently more tractable. The problem we study differs from the original question in two ways. First, we are considering the list version of the problem: in an input instance of the *list homomorphism* LHom(H) problem, each vertex v of G is equipped with a list $L(v) \subseteq V(H)$ and the task is to find a homomorphism ϕ from G to H that respects these lists, that is, $\phi(v) \in L(v)$ for every $v \in V(G)$. List versions of homomorphism and coloring problems are well studied [24, 7, 8, 13, 15, 12, 11, 14, 10, 9]. Typically, list versions are more robust than the ordinary versions and hardness proofs are simpler to prove for them. Feder et al. [10] characterized the polynomial-time solvable cases of LHom(H): now it is not sufficient that H is bipartite, it has to be the complement of a circular arc graph, otherwise the problem is NP-complete. Second, we consider the *reflexive* version of the homomorphism problem, which means that we assume that every vertex of H has a self-loop attached to it. Thus even if u and v are adjacent in G, it is still possible that $\phi(u) = \phi(v)$ in a homomorphism ϕ from G to H. In particular, now there is always a homomorphism ϕ from every G to H: let us chose an arbitrary fixed vertex $u \in V(H)$ and let $\phi(v) = u$ for every $v \in V(G)$. However, it remains a nontrivial question whether there is a homomorphism from G to H that respects the lists $L(v)$ of the vertices of G. Feder and Hell [9] showed that reflexive LHom(H) is polynomial-time solvable if H is an interval graph, and NP-complete otherwise. In general, the reflexive problem appears to have simpler structure and cleaner properties than the irreflexive version, where bipartiteness and parity issues introduce technical complications. We believe that it is reasonable to start with the reflexive problem as a prototype result.

Results. Our main result is exactly characterizing, for every fixed H, the complexity of reflexive list homomorphism parameterized by treewidth. Similarly to the c-Coloring problem, standard dynamic programming techniques give an algorithm with running time $|V(H)|^{\mathrm{tw}(G)} \cdot n^{O(1)}$ if a tree decomposition of width $\mathrm{tw}(G)$ is given (slightly more generally, we can also say that if every list $L(v)$ has size at most c, then the problem can be solved in time $c^{\mathrm{tw}(G)} \cdot n^{O(1)}$). However, unlike in the case of the c-Coloring problem, this algorithm is not necessarily optimal: for some H, we can actually do better. We identify two algorithmic ideas that can give improved algorithms:

- **Incomparable sets.** Suppose that the list $L(v)$ for some $v \in V(G)$ contains two vertices $a, b \in V(H)$, such that every neighbor of a (including a itself) is also a neighbor of b. It is easy to see that if there is a homomorphism ϕ from G to H with $\phi(v) = a$, then this can be modified to have $\phi(v) = b$ and it remains a valid homomorphism. In other words, vertex a in the list $L(v)$ is not necessary for the solution and can be removed from the list. Thus after a simple preprocessing step, we can assume that every $L(v)$ is an *incomparable set,* that is, $N[a] \subseteq N[b]$ does not hold for any two distinct $a, b \in L(v)$. This means that if we denote by $i(H)$ the maximum size of an incomparable set in H, then it can be assumed that every list has size at most $i(H)$ and hence the problem can be solved in time $i(H)^{\mathrm{tw}(G)} \cdot n^{O(1)}$. As $i(H)$ can be much less than $|V(H)|$, this running time can be significantly faster than $|V(H)|^{\mathrm{tw}(G)} \cdot n^{O(1)}$.
- **Decompositions.** We identify a certain kind of decomposition that can be used to simplify the problem. Formally, a decomposition is a partition (S, N, R) of the vertices of H such that $|S| \geq 2$, $|N \cup R| > 0$, N separates S and R, N induces a clique, and every vertex of S is adjacent to every vertex of N. As we show later, such a decomposition allows us to reduce LHom(H) to instances of LHom(H_1) and LHom(H_2), where H_1 and H_2 are strict induced subgraphs of H.

We show that, in a formal sense, these two algorithmic ideas are sufficient to solve the problem as fast as possible. First, if the graph H is *undecomposable* (that is, does not have a decomposition as above), then the best possible running time is indeed of the form $i(H)^{\mathrm{tw}(G)} \cdot n^{O(1)}$. More generally, we define $i^*(H)$ to be the maximum of $i(H^*)$, taken over every undecomposable, connected, non-interval induced subgraph H^* of H. Our main result shows that $i^*(H)^{\mathrm{tw}(G)} \cdot n^{O(1)}$ is the exact complexity of the problem.

▶ **Theorem 2.** *Let H be a connected reflexive non-interval graph with $k = i^*(H)$, and G be a graph with n vertices and treewidth* $\mathrm{tw}(G)$*.*

(a) *The* LHom(H) *problem with instance (G, L) can be solved in time poly$(n + |H|) \cdot k^{\mathrm{tw}(G)}$ for any lists L, provided that G is given with its tree decomposition of width* $\mathrm{tw}(G)$*.*

(b) *There is no algorithm that solves* LHom(H) *for every G and L in time $f(H) \cdot$ poly$(n + |H|) \cdot (k - \epsilon)^{\mathrm{tw}(G)}$ for any computable function f and any $\epsilon > 0$, unless the SETH fails.*

Note that if H is a reflexive interval graph, then LHom(H) is polynomial-time solvable [9] and if H is disconnected, then it is easy to reduce the problem to the components of H. Thus Theorem 2 gives a complete characterization of the complexity of the problem for every fixed H.

Let us discuss the significance of a complete classification result such as Theorem 2. As the LHom(H) problem is an infinite family of problems, it is not clear at all what is the full range of algorithmic ideas that can help solve the problem faster than the naive $|V(H)|^{\mathrm{tw}(G)} \cdot n^{O(1)}$ time algorithm. Even after realizing that this naive algorithm can be beaten in some cases (e.g., by discovering the importance of incomparable sets or some form of decompositions), we cannot be sure that some completely different algorithm cannot solve some cases even faster, or can be applied to an even wider class of target graphs H. But in order to prove a complete classification result of the form of Theorem 2, one has to discover each and every relevant algorithmic idea. Our main result not only provides a set of algorithmic tools, but proves in a formal sense (assuming the SETH) that no other algorithmic idea can improve on these results. Thus we completely map the complexity landscape of the LHom(H) problem, determining the complexity of every case of LHom(H) with surprising tightness and revealing every combinatorial insight that can be exploited algorithmically.

Lower bound proofs. The complexity result of Theorem 2b needs to exploit three properties of the induced subgraph H^*: it is not an interval graph, it is undecomposable, and has a large incomparable set. There are well-known characterization results that show that every non-interval graph contains certain obstructions (induced cycles or asteroidal triples) and the NP-hardness proofs of Feder and Hell [9] show how these obstructions can be used to reduce 3-Coloring to LHom(H). However, here we need something much stronger: if there is an incomparable set I of size $c = i^*(H)$, then we want to reduce c-Coloring to LHom(H) and use the lower bound in Theorem 1. The natural idea is to represent the c colors of the c-Coloring problem by the c vertices appearing in the incomparable set I. Then the main challenge is to construct gadgets that express the $\neq$ relation, that is, ensure that two adjacent vertices are not assigned the same vertex of I, but every other combination is allowed. We show with a very delicate and technical proof that the incomparable set can be connected to the interval graph obstruction with a set of walks satisfying certain properties, and these walks, together with the obstruction, can be used to create the required gadgets. It turns out that, surprisingly, the only situation when we cannot find such walks is precisely when a decomposition exists. Thus if we assume that the graph is non-interval, has a large incomparable set, and has no decomposition, then we can construct the gadgets required for the reduction.

Exploiting decompositions. We finish the introduction with a brief explanation of how a decomposition (S, N, R) can be exploited (a more detailed algorithm description appears in

Section 3.1). As discussed above, we can assume that every $L(v)$ is an incomparable set in H. In particular, this means that if some $a \in S$ appears in $L(v)$, then $L(v)$ does not contain any vertex of N (as every vertex of N fully contains the neighborhood of every vertex in S). Let $X \subseteq V(G)$ be the set of vertices whose lists contain at least one vertex of S. The set X induces some number of connected components in G; let C be a connected component of $G[X]$.

The first crucial observation is that in every solution ϕ, one of the following two cases has to happen on C: either (1) $\phi(c) \in S$ for every $c \in C$, or (2) $\phi(c) \notin S$ for every $c \in C$. Otherwise, there would be two adjacent vertices $c_1, c_2 \in C$ with $\phi(c_1) \in S$ and $\phi(c_2) \notin S$. However, as N separates S and R in H, this is only possible if $\phi(c_2) \in N$, contradicting our earlier assumption. Thus as a first step, we check if there is a homomorphism ϕ_C from $G[C]$ to $H_1 := H[S]$ that respects the list. If there is no such homomorphism, then we can rule out the possibility that case (1) happens on C and remove the vertices of S from the lists of the vertices in C. Suppose now that there is such a homomorphism ϕ_C.

The second crucial observation is that if case (1) happens on C, then we might as well assume that the solution ϕ restricted to C is exactly the same as ϕ_C: it is easy to see that $\phi(v) \in N$ should hold for every $v \in N(C)$, and every vertex of N is adjacent to every vertex of S, hence no conflict can arise if we change ϕ to be the same as ϕ_C on C. Let us select an arbitrary vertex $a \in S$ and let us change the list of every $v \in C$ to be $L'(v) = (L(v) \setminus S) \cup \{a\}$, that is, the single vertex a will represent the vertices $L(v) \cap S$. We claim that this modification does not change the solvability of the instance: if the original instance has a solution where case (1) happens on C, then we can modify it to have a's on every vertex of C; and if we obtain a solution of the new instance with a's on C, then we can obtain a solution of the original instance by using ϕ_C on C.

We repeat these steps for every connected component C of $G[X]$. Then we obtain an instance where the selected vertex $a \in S$ is the only vertex of S that appears anywhere on the lists. This means that effectively we have an instance where we need to find a homomorphism to $H_2 := H \setminus (S \setminus \{a\})$. As $|S| \geq 2$, H_2 has strictly fewer vertices than H. Thus the existence of the decomposition (S, N, R) allowed us to reduce the problem to instances of LHom(H_1) and LHom(H_2) where H_1 and H_2 have fewer vertices than H.

2 Preliminaries

Throughout the paper we consider reflexive graphs only, i.e., we assume that for every vertex v, vv is an edge (a loop). Let $H = (V, E)$ be a reflexive graph. By $N[v]$ we denote the set $\{u \colon uv \in E\}$. Note that $v \in N[v]$. By $N(v)$ we denote $N[v] \setminus \{v\}$. For a set X of vertices, by $N[X]$ we denote $\bigcup_{v \in X} N[v]$, while $N(X)$ denotes $N[X] \setminus X$. For a set X and a vertex v, by $N_X[v]$ we denote $N[v] \cap X$. In an analogous way we define $N_X(v)$ and $N_X(Y)$ and $N_X[Y]$ for a set Y. For two disjoint sets $A, B \subseteq V$, such that no vertex from A is adjacent to a vertex from B, an A-B-separator is a set S, such that there is no path from any vertex $a \in A$ to any vertex $b \in B$ in the graph $H - S$. An A-B-separator S is *minimal* if no $S' \subsetneq S$ is an A-B-separator. If A is a singleton, say $A = \{a\}$, we write a-B-separator instead of $\{a\}$-B separator (analogously if B is a singleton).

For two graphs G and H, a mapping $f \colon V(G) \to V(H)$ is a *homomorphism* if for every edge xy of G it holds that $f(x)f(y)$ is an edge of H. If f is a homomorphism from G to H, we denote it shortly by $f \colon G \to H$. We write $G \to H$ to say that there exists some homomorphism from G to H. For a fixed graph H, by Hom(H) we consider an algorithmic problem of deciding if there is a homomorphism from a given graph G to H.

In the *list homomorphism* problem we are given two graphs G, H and a mapping $L\colon V(G) \to 2^{V(H)}$ (where the sets $L(v)$ for $v \in V(G)$ are called *H-lists* or just *lists*), and we ask for a homomorphism $f\colon G \to H$, such that $f(x) \in L(x)$ for every $x \in V(G)$. We denote this by $f\colon (G,L) \to H$. We often write $(G,L) \to H$ to denote that there is a list homomorphisms from G to H with lists L. Moreover, as we only deal with list homomorphisms, we write $f\colon G \to H$ to denote $f\colon (G,L) \to H$, if the lists are clear from the context. For a fixed graph H, by LHom(H) we denote the algorithmic problem, whose input is a graph G with lists L, and we ask whether there exists a list homomorphism $(G,L) \to H$.

We observe that if H has several connected components, then there is a polynomial-time reduction from LHom(H) to the problems LHom(H') for the connected components H' of H. Thus we always assume that H is connected.

2.1 Interval graphs and obstructions

Interval graphs are one of the most studied classes of geometric intersection graphs. A graph H is an interval graph if it admits an *interval representation*, where each vertex is represented by some closed interval of the real line and two vertices are adjacent if and only if their corresponding intervals intersect. Note that interval graphs are usually defined to be irreflexive, but in our case we consider reflexive graphs.

Before we analyze structural properties of interval graphs, we need a few more definitions. An *asteroidal triple* is an independent set of three vertices a, b, c, such that for every $\{i, j, \ell\} = \{a, b, c\}$ there is an i-j-path $\mathcal{W}_{i,j}$, whose every vertex is non-adjacent to ℓ. Note that by our convention $\mathcal{W}_{j,i}$ is $\mathcal{W}_{i,j}$ reversed.

▶ **Theorem 3** (Lekkeikerker and Boland [19]). *A graph is an interval graph if and only if does not contain an asteroidal triple or an induced cycle of length at least 4.*

There is a deep connection between the list homomorphism problem and reflexive interval graphs, as shown in the following dichotomy theorem of Feder and Hell [9].

▶ **Theorem 4** (Feder and Hell [9]). *Let H be a reflexive graph. If H is an interval graph, then the* LHom(H) *problem is polynomially solvable, otherwise it is NP-complete.*

In this paper we will focus on graphs H, for which LHom(H) is NP-complete, so we will assume that H is non-interval and thus contains at least one of structures mentioned in Theorem 3.

Observe that for an asteroidal triple a, b, c, we may w.l.o.g. assume that each path $\mathcal{W}_{a,b}$ is induced. We define the *asteroidal subgraph* of an asteroidal triple a, b, c, as the subgraph of H induced by $\mathcal{W}_{a,b} \cup \mathcal{W}_{b,c} \cup \mathcal{W}_{a,c}$.[1] For a vertex a (b, c, resp.), we say that the path $\mathcal{W}_{b,c}$ ($\mathcal{W}_{a,c}$, $\mathcal{W}_{a,b}$, resp.) is *opposite.*

Moreover, note that an induced cycle with at least 6 vertices contains an asteroidal subgraph. So an equivalent statement of Theorem 3 says that every non-interval graph H contains an induced 4-cycle, an induced 5-cycle, or an asteroidal subgraph. An induced subgraph of H isomorphic to one of these three structures is called an *obstruction* in H.

A vertex $o \in \mathbb{O}$ is a *corner* if:

- $\mathbb{O}$ is isomorphic to a 4-cycle or a 5-cycle, or
- $\mathbb{O}$ is an asteroidal subgraph for an asteroidal triple containing o.

Two vertices o, o' of an obstruction $\mathbb{O}$ are *opposite* if:

[1] We will often identify graphs with their vertex sets.

- both are corners, or
- $\mathbb{O}$ is an asteroidal subgraph and o belongs to the path opposite to o'.

2.2 Dominating vertices and incomparable sets

For two vertices u, v of H, we say that v *dominates* u or, equivalently, u is *dominated* by v, if $N[u] \subseteq N(v)$. Observe that this implies that u and v are adjacent. We say that a set X *dominates* a set Y if every $x \in X$ dominates every $y \in Y$. If u is not dominated by v, it means that there is a vertex $u' \in N[u]$ (possibly $u' = u$), which is not a neighbor of v. Two vertices u and v are *incomparable* if u does not dominate v and v does not dominate u. A set S of vertices is *incomparable* if all its members are pairwise incomparable. By $i(H)$ we denote the size of the largest incomparable set in H.

2.3 Avoiding walks

A *walk* is a sequence of vertices $\mathcal{P} = p_1, p_2, \ldots, p_\ell$, such that $p_i p_{i+1}$ is an edge for every $i = 1, 2, \ldots, \ell - 1$. For the walk $\mathcal{P}$, its *length* denotes the number $\ell - 1$. For two vertices a, b, we say that $\mathcal{P} = p_1, p_2, \ldots, p_\ell$ is an a-b-walk if $p_1 = a$ and $p_\ell = b$. We denote this shortly by $\mathcal{P} \colon a \to b$. By $\bar{\mathcal{P}}$ we denote the *reversed walk*, i.e., $\bar{\mathcal{P}} = p_\ell, p_{\ell-1}, \ldots, p_2, p_1$.

For two walks $\mathcal{A} = a_1, a_2, \ldots, a_\ell$ and $\mathcal{B} = b_1, b_2, \ldots, b_{\ell'}$ such that $a_\ell = b_1$, we let $\mathcal{A} \circ \mathcal{B}$ denote the *concatenation* of $\mathcal{A}$ and $\mathcal{B}$, i.e., the walk $a_1, a_2, \ldots, a_\ell, b_2, b_3, \ldots, b_{\ell'}$. Note that $|\mathcal{A} \circ \mathcal{B}| = |\mathcal{A}| + |\mathcal{B}| - 1$.

For two walks $\mathcal{P} = p_1, p_2, \ldots, p_\ell$ and $\mathcal{Q} = q_1, q_2, \ldots q_\ell$ of equal length, we say that $\mathcal{P}$ *avoids* $\mathcal{Q}$ if p_i is non-adjacent to q_{i+1} for every $i = 1, 2, \ldots, \ell - 1$. We conclude this section with two simple observations concerning walks and avoidance.

▶ **Observation 5.** *For walks $\mathcal{A} \colon a \to b$, $\mathcal{B} \colon b \to c$ and $\mathcal{A}' \colon a' \to b', \mathcal{B}' \colon b' \to c'$, if $\mathcal{A}$ avoids $\mathcal{A}'$ and $\mathcal{B}$ avoids $\mathcal{B}'$, then $\mathcal{A} \circ \mathcal{B}$ avoids $\mathcal{A}' \circ \mathcal{B}'$.* ◀

▶ **Observation 6.** *Let $\mathcal{P} = p_1, p_2, \ldots, p_\ell$ and $\mathcal{Q} = q_1, q_2, \ldots q_\ell$ be two walks, such that $\mathcal{P}$ avoids $\mathcal{Q}$. Then $\bar{\mathcal{Q}}$ avoids $\bar{\mathcal{P}}$.* ◀

3 Algorithm

In this section we prove the algorithmic part of our main result, i.e., Theorem 2a). Let us start with the following simple observation.

▶ **Observation 7.** *Let u, v be vertices of H, such that v dominates u. Let $f \colon G \to H$ be a homomorphism, such that $f(x) = u$ for some vertex x of G. Then f' defined by $f'(x) := v$ and $f'(y) := f(y)$ for every $y \in V(G) \setminus \{x\}$ is also a homomorphism from G to H.* ◀

Thus we can assume that in our instance (G, L) of LHom(H) the set $L(x)$ is incomparable for every vertex x of G (otherwise we can safely remove a dominated vertex).

3.1 Decomposition

For a graph H, let $T(H, n, t)$ denote an upper bound for the time complexity of an algorithm solving the LHom(H) problem on a graph with n vertices and treewidth t. The following lemma is the main tool in the proof of Theorem 2a). The proof is omitted in this extended abstract.

▶ **Lemma 8** (Decomposition lemma). *Let $H = (V, E)$ be a reflexive graph, whose vertex set can be partitioned into three subsets S, N, R, such that:*

1. $|S| \geq 2$,
2. *N is a clique with at least one vertex,*
3. *N separates S and R,*
4. *all edges between S and N are present in H.*

Let H_1 be the subgraph of H induced by S, and H_2 be the subgraph of H obtained by contracting S to a single vertex. Moreover, suppose that there are constants c, d, such that $T(H_1, n, t) = O(c^t \cdot n^d)$ and $T(H_2, n, t) = O(c^t \cdot n^d)$. Then $T(H, n, t) = O(c^t \cdot n^d)$.

A graph H which satisfies the assumptions of Lemma 8 is called *decomposable* and we say that (S, N, R) is a *decomposition* of H, or that H *decomposes* into H_1 and H_2. We refer to S as *dominated part* and the set N as *dominating clique separator.* A graph which is not decomposable is called *undecomposable.*

Observe that with H we can associate a *decomposition tree* $\mathcal{T}$, whose nodes are labeled with induced subgraphs of H. The root, denoted by *node*(H) corresponds to the whole graph H. If H is undecomposable, then the decomposition tree has just one node. If H decomposes into H_1 and H_2, then *node*(H) has two children, *node*(H_1) and *node*(H_2), respectively. We construct a decomposition tree recursively. Clearly, each leaf of the decomposition tree is an undecomposable induced subgraph of H. Note that a decomposition tree may not be unique, as a graph may have more than one decomposition. However, the number of leaves is always $O(|H|)$, so the total number of nodes is also $O(|H|)$.

3.2 Solving LHom(H) problem

Now we are ready to present an algorithm for determining if $(G, L) \to H$.

Proof of Theorem 2a). We assume that the graph G has n vertices and is given along with its tree decomposition of width $\mathrm{tw}(G)$. We also define

$$i^*(H) := \max\{i(H') \colon H' \text{ is undecomposable connected non-interval induced subgraph of } H\}.$$

Observe that if H' is an induced subgraph of H, then $i^*(H') \leq i^*(H)$, and thus $i(H) = i^*(H)$ for undecomposable H.

It can be shown that in time polynomial in H we can check if H is undecomposable, or find a decomposition. If H is undecomposable, we run a standard dynamic programming on a tree decomposition of G (see [3, 2]). For each bag of the tree decomposition we store all partial list homomorphisms from the graph induced by this bag to H. By Observation 7, the size of each list $L(x)$ for $x \in V(G)$ is at most $i(H)$, thus the complexity of the dynamic programming algorithm is bounded by $O(n^d \cdot i(H)^{\mathrm{tw}(G)}) = O(n^d \cdot i^*(H)^{\mathrm{tw}(G)})$ for some constant d.

So suppose H is decomposable. Let $\mathcal{T}$ be a decomposition tree of H, note that it can be constructed in polynomial time, has $O(|H|)$ nodes, and its every leaf corresponds to an induced subgraph of H with strictly fewer vertices. Indeed, if H decomposes into H_1 and H_2, then they are both induced subgraphs of H and $|H_1|, |H_2| < |H|$. Therefore, for any leaf H' of $\mathcal{T}$, we can solve every instance of LHom(H') with n vertices and treewidth at most $\mathrm{tw}(G)$ in time $O(n^d \cdot i^*(H)^{\mathrm{tw}(G)})$ (note that this is also true if H' is an interval graph, as then we can use a polynomial algorithm). Now, applying Lemma 8 in a bottom-up fashion, we conclude that we can solve LHom(H) in time $O(n^d \cdot i^*(H)^{\mathrm{tw}(G)})$, which completes the proof of Theorem 2a). ◀

4 Hardness

In this section we prove Theorem 2b), i.e., the lower bound for an algorithm deciding the existence of a list homomorphism $(G, L) \to H$. We will prove the following theorem.

▶ **Theorem 9.** *Let H be a connected, reflexive, undecomposable graph with $i(H) \geq 3$. Assuming the SETH, there is no algorithm that solves* LHom*(H) for every G and L in time $f(H) \cdot poly(|G| + |H|) \cdot (i(H) - \epsilon)^{\mathrm{tw}(G)}$ for any $\epsilon > 0$ and any computable function f.*

Let us first show that Theorem 9 is equivalent to Theorem 2b).

Theorem 9 $\to$ Theorem 2b). Suppose Theorem 9 holds and Theorem 2b) fails. So there is a graph H (may be decomposable) and an algorithm A that solves LHom(H) in time $f(H) \cdot poly(|G| + |H|) \cdot (i^*(H) - \epsilon)^{\mathrm{tw}(G)}$ for every input G, L. Let H' be an undecomposable connected non-interval induced subgraph of H, such that $i(H') = i^*(H)$. As every instance of LHom(H') can be seen as an instance of LHom(H), the algorithm A can be used to solve LHom(H') in time $f'(H') \cdot poly(|G| + |H'|) \cdot (i(H') - \epsilon)^{\mathrm{tw}(G)}$, thus contradicting Theorem 9.

Theorem 2b) $\to$ Theorem 9. Suppose Theorem 2b) holds and Theorem 9 fails. So there is an undecomposable graph H and an algorithm A that solves LHom(H) in time $f(H) \cdot poly(|G| + |H|) \cdot (i(H) - \epsilon)^{\mathrm{tw}(G)}$ for every input G, L. But since H is undecomposable, we have $i^*(H) = i(H)$, so algorithm A contradicts Theorem 2b).

4.1 Using an obstruction to express basic relations

Let $\mathbb{O}$ be an obstruction in H with non-adjacent corners α, β and let $k \geq 2$ be an integer. First, we show how express k-wise relations $OR_k = \{\alpha, \beta\}^k \setminus \alpha^k$ and $NAND_k = \{\alpha, \beta\}^k \setminus \beta^k$. More formally, we define a graph $F(OR_k)$ ($F(NAND_k)$, resp.), called an *OR_k-gadget* (*$NAND_k$-gadget*, resp.) with H-lists L and k specified vertices $x_1, x_2, \ldots, x_k$, such that:

- for every $i \in [k]$ it holds that $L(x_i) = \{\alpha, \beta\}$,
- the relation $\bigcup_{f\colon F(OR_k) \to H} \{f(x_1)f(x_2)\ldots f(x_k)\}$ is exactly OR_k (respectively, $\bigcup_{f\colon F(NAND_k) \to H} \{f(x_1)f(x_2)\ldots f(x_k)\}$ is $NAND_k$).

The construction of these gadgets is simple and it is omitted in this extended abstract. Another useful property of obstructions is shown in the following lemma.

▶ **Lemma 10** (Moving inside the obstruction). *Let $\mathbb{O}$ be an obstruction with distinct corners a, c. Moreover, let b, d be distinct vertices of $\mathbb{O}$, such that b is a corner and d is either a corner, or a vertex non-adjacent to b. Then there are walks $\mathcal{A}_{a,b}, \mathcal{A}'_{a,b} : a \to b$ and $\mathcal{B}_{c,d}, \mathcal{B}'_{c,d} : c \to d$, such that $\mathcal{A}_{a,b}$ avoids $\mathcal{B}_{c,d}$ and $\mathcal{B}'_{c,d}$ avoids $\mathcal{A}'_{a,b}$. Moreover, all four walks use only vertices of $\mathbb{O}$ and can be constructed in polynomial time.*

Proof. If $\mathbb{O}$ is an induced 4-cycle or an induced 5-cycle, the walks are easy to construct. So consider the case that $\mathbb{O}$ is an asteroidal subgraph for an asteroidal triple o_1, o_2, o_3.

Case 1. First, let us deal with case when both b, d are corners. Then we have $\{a, b, c, d\} \subseteq \{o_1, o_2, o_3\}$. If $a = b$ and $c = d$ then the problem is trivial. If $a = b$ and $c \neq d$, then we set $\mathcal{A}_{a,b} = \mathcal{A}'_{a,b} = a, a, \ldots, a$ and $\mathcal{B}_{c,d} = \mathcal{B}'_{c,d} = \mathcal{W}_{c,d}$ (the walk opposite to a). The case when $a \neq b$ and $c = d$ is similar. So we assume that $a \neq b$ and $c \neq d$. If $a = d$ and $c \neq b$ (the case when $a \neq d$ and $c = b$ is similar), we set $\mathcal{A}_{a,b} = \mathcal{A}'_{a,b} = \mathcal{W}_{a,b} \circ b, b, \ldots, b$ and $\mathcal{B}_{c,d} = \mathcal{B}'_{c,d} = c, c, \ldots, c \circ \mathcal{W}_{c,a}$. If $a = d$ and $c = b$, we set $\mathcal{A}_{a,b} = \mathcal{A}'_{a,b} = a, a, \ldots, a \circ \mathcal{W}_{a,c} \circ c, c, \ldots, c$ and $\mathcal{B}_{c,d} = \mathcal{B}'_{c,d} = \mathcal{W}_{c,b} \circ b, b, \ldots, b \circ \mathcal{W}_{b,a}$.

Case 2. Next, consider the case when b is a corner and d is not. We know that $d \in \mathcal{W}_{b',b''}$, where b', b'' are corners and $b' \neq b$. Let $\mathcal{W}'$ be the subpath of $\mathcal{W}_{b',b''}$, starting in b' and ending in d. Recall that $\mathcal{W}_{b',b''}$ is induced, so even if $b = b''$, there is no edge from $\mathcal{W}'$ to b. We set

$$\begin{array}{lllllll} \mathcal{A}_{a,b} = & \mathcal{C}_{a,b} & \circ & b, b, \ldots, b & \mathcal{A}'_{a,b} = & \mathcal{C}'_{a,b} & \circ \quad b, b, \ldots, b \\ \mathcal{B}_{c,d} = & \mathcal{D}_{c,b'} & \circ & \mathcal{W}' & \mathcal{B}'_{c,d} = & \mathcal{D}'_{c,b'} & \circ \quad \mathcal{W}', \end{array}$$

where $\mathcal{C}_{a,b}, \mathcal{C}'_{a,b} \colon a \to b$ and $\mathcal{D}_{c,b'}, \mathcal{D}'_{c,b'} \colon c \to b'$ are appropriate walks given by Case 1. ◀

4.2 Constructing distinguishing walks

As we have seen, we can easily use the structure of an obstruction to enforce non-trivial relations that could be used to show hardness. For the rest of the proof we will show that we can attach vertices of an incomparable set to the vertices of an obstruction using walks with certain avoidance properties, which will later be exploited to prove hardness.

For a walk $\mathcal{P} = v_1, v_2, \ldots, v_n$, by $\widetilde{\mathcal{P}}$ we denote the walk $\mathcal{P}$ with its first vertex removed, i.e., $\widetilde{\mathcal{P}} = v_2, \ldots, v_n$. The following structural lemmas will be later used to obtain the main gadget used in our hardness proof.

▶ **Lemma 11.** *Let H be a connected undecomposable non-interval reflexive graph, and $\mathbb{O}$ be an obstruction in H with non-adjacent corners α, β. Let $\mathcal{S}$ be a set of incomparable vertices in H such that $|\mathcal{S}| \geq 2$. Let a and b be arbitrary distinct vertices in $\mathcal{S}$. Then there is a partition (X, Y) of $\mathcal{S}$ such that vertices a and b are in X, and there are walks $\mathcal{D}_v$ for each $v \in \mathcal{S}$ of length at least 1, satisfying the following properties:*

1. *For each $v \in \mathcal{S}$, the first vertex of $\mathcal{D}_v$ is v, and its last vertex is either α ($\mathcal{D}_v$ is said to be an α-walk) or β ($\mathcal{D}_v$ is said to be a β-walk).*
2. *$\mathcal{D}_a$ is an α-walk and $\mathcal{D}_b$ is a β-walk.*
3. *Let $u, v \in \mathcal{S}$ such that $\mathcal{D}_u$ is an α-walk and $\mathcal{D}_v$ is a β-walk. Then*
   a. *if $u, v \in X$, or if $u \in Y$ and $v \in X \cup Y$, then $\mathcal{D}_u$ avoids $\mathcal{D}_v$,*
   b. *if $u \in X$, $v \in Y$, then $\widetilde{\mathcal{D}}_u$ avoids $\widetilde{\mathcal{D}}_v$.*
4. *For any $v \in Y$ and $u \in X$, there is no edge joining v and the second vertex of $\mathcal{D}_u$.*
5. *For every $v \in Y$, the second vertex of $\mathcal{D}_v$ is in Y.*

Recall that it is possible that we have two walks $\mathcal{D}_x, \mathcal{D}_y$ constructed in Lemma 11, such that $\mathcal{D}_x$ is an α-walk, $\mathcal{D}_y$ is a β-walk, but $\mathcal{D}_x$ **does not** avoid $\mathcal{D}_y$ (this may happen for $x \in X$ and $y \in Y$). This is an undesired situation for us, but luckily such walks have a well-defined structure. In the next lemma we will construct a small gadget to patch this situation, and then we will combine them to construct the main tool in our hardness proof, i.e., a *distinguisher gadget*.

▶ **Lemma 12.** *Let $H, \mathcal{S}, X, Y$, and $\mathcal{D}_v$, where $v \in \mathcal{S}$, be as in Lemma 11. Let $N_X = \{d_2^x \colon x \in X\}$, i.e., the set of vertices that appear as a second vertex of a walk $\mathcal{D}_x$ where $x \in X$ and $N_Y = \{d_2^y \colon y \in Y\}$. Then there is a graph F with H-lists and two specified vertices $p_1, p_2 \in V(F)$ such that*

1. *$L(p_1) = \mathcal{S}$ and $L(p_2) = N_X \cup N_Y$,*
2. *for any list homomorphism $\varphi \colon F \to H$, if $\varphi(p_1) \in X$, then $\varphi(p_2) \notin Y$,*
3. *for every $v \in \mathcal{S}$, there is $\psi \colon F \to H$, such that $\psi(p_1) = v$ and $\psi(p_2) = d_2^v$.*

4.3 Constructing a distinguisher gadget

The main tool in our hardness proof is a gadget called a *distinguisher*. Let H be an undecomposable reflexive non-interval graph with obstruction $\mathbb{O}$ with non-adjacent corners α, β. For an incomparable set $\mathcal{S}$ of H and two vertices $a, b \in \mathcal{S}$, a distinguisher is a graph $D_{a/b}$ with two specified vertices x, y and H-lists L, such that:

1. $L(x) = \mathcal{S}$ and $L(y) = \{\alpha, \beta\}$,
2. there is a list homomorphism $\phi_a \colon D_{a/b} \to H$, such that $\phi_a(x) = a$ and $\phi_a(y) = \alpha$,
3. there is a list homomorphism $\phi_b \colon D_{a/b} \to H$, such that $\phi_b(x) = b$ and $\phi_b(y) = \beta$,
4. for any $c \in \mathcal{S} \setminus \{a, b\}$ there is $\phi_c \colon D_{a/b} \to H$, such that $\phi_c(x) = c$ and $\phi_c(y) \in \{\alpha, \beta\}$,
5. there is no list homomorphism $\phi \colon D_{a/b} \to H$, such that $\phi(x) = a$ and $\phi(y) = \beta$.

▶ **Lemma 13** (Construction of distinguisher). *Let $H = (V, E)$ be an undecomposable reflexive non-interval graph with obstruction $\mathbb{O}$ with two non-adjacent corners α, β. Let $\mathcal{S}$ be a maximum incomparable set in H. Then for every ordered pair (a, b) of distinct elements of $\mathcal{S}$ there exists a distinguisher $D_{a/b}$.*

Proof. Call Lemma 11 for $H, \mathcal{S}, a, b$ to obtain a partition (X, Y) of $\mathcal{S}$ and walks $\mathcal{D}_v$ for every $v \in \mathcal{S}$. Let s be the length of each of these walks. By d^v_j we denote the j-th vertex of $\mathcal{D}_v$.

Let P be a path with s vertices $p_1, p_2, \ldots, p_s$. We set $L(p_j) = \bigcup_{v \in \mathcal{S}} \{d^v_j\}$. Observe that by Lemma 11 we have $s \geq 2$.

Next, call Lemma 12 to obtain a graph F_P and unify its p_1-vertex with p_1 of P and its p_2-vertex with p_2 of P. Observe that this unification preserves lists. Finally, we set $x = p_1$ and $y = p_s$. Let us verify that the graph constructed in such a way is indeed a distinguisher. The first property holds by the definition of the walks $\mathcal{D}_v$ for $v \in \mathcal{S}$. To show properties 2,3, and 4, consider $v \in \mathcal{S}$ and set $\phi_v(p_i) = d^v_i$ for all $i \in [s]$. This mapping can be extended to the vertices of F_P by property 3 of Lemma 12.

Finally, let us show that property 5 holds as well. Assume for the sake of contradiction that a list homomorphism $\phi \colon D_{a/b} \to H$, such that $\phi(x) = a$ and $\phi(y) = \beta$, exists. Observe that $\phi(p_1), \phi(p_2), \ldots, \phi(p_s)$ is an a-β walk of length s in H, such that for every $i \in [s]$ we have $\phi(p_i) \in \bigcup_{v \in \mathcal{S}} \{d^v_2\}$. For all $i \in [s]$, let $\mathcal{D}^i$ denote a walk from $\{\mathcal{D}_v \colon v \in \mathcal{S}\}$, whose i-th vertex is $\phi(p_i)$ (if there is more than one such a walk, choose an arbitrary one). Observe that $\mathcal{D}^1 = \mathcal{D}_a$ is an α-walk. Let i be a minimum integer, such that $\mathcal{D}^i$ is a β-walk. This value is well-defined, as $\mathcal{D}^s$ is a β-walk. Thus there is an edge between the $(i-1)$th vertex of the α-walk $\mathcal{D}^{i-1}$ and the i-th vertex of the β-walk $\mathcal{D}^i$, so $\mathcal{D}^{i-1}$ does no avoid $\mathcal{D}_i$. Let u, v be vertices such that $\mathcal{D}^{i-1} = \mathcal{D}_u$ and $\mathcal{D}^i = \mathcal{D}_v$. If $u, v \in X$, or $u \in Y$ and $v \in X \cup Y$, then we have a contradiction with property 3a in Lemma 11.

Thus assume that $u \in X$ and $v \in Y$. If $i \geq 3$, then again we have a contradiction with property 3b in Lemma 11. Thus the only case left is $i = 2$. By property 2. in Lemma 12 we observe that $d^v_2 \notin Y$, and thus, by property 5 in Lemma 11, we conclude that $v \notin Y$, a contradiction. This completes the proof. ◀

4.4 Hardness proof

We are ready to prove to prove Theorem 9. Recall that Theorem 9 implies Theorem 2b).

Proof of Theorem 9. Let $\mathbb{O}$ be an obstruction in H and let α, β be non-adjacent corners of $\mathbb{O}$. Let $\mathcal{S} = \{v_1, v_2, \ldots, v_k\}$ be a maximum incomparable set in H. Note that we can assume that $k \geq 3$, since the corners of $\mathbb{O}$ are pairwise incomparable.

Suppose we are given a graph G along with its tree decomposition of width $\mathrm{tw}(G)$. The main idea of our hardness proof is to construct a graph G^* with H-lists L such that:

- $(G^*, L) \to H$ if and only if G is k-colorable (in our construction the colors used on G will correspond to vertices from $\mathcal{S}$),
- the number of vertices of G^* is $g(H) \cdot (|V(G)| + |E(G)|)$ for some function g of H,
- the treewidth of G^* is at most $g(H) + \text{tw}(G)$,
- G^* can be constructed in time $poly(|V(G)|) \cdot g'(H)$ for some function g'.

Invoking Theorem 1, this will prove Theorem 9. The construction is performed in four steps.

Step 1. Constructing an indicator gadget. Fix $i \in [k]$. For $j \in [k] \setminus \{i\}$, we use Lemma 13 to construct a distinguisher gadget D_{v_i/v_j} with two specified vertices $x_{i,j}$ and $y_{i,j}$. The number of constructed gadgets is thus $k-1$. We identify vertices $x_{i,j}$ for all $j \in [k]$, let us call this identified vertex x_i. Moreover, introduce a new vertex c_i. Now, using the construction from Section 4.1 we introduce an OR_k gadget and identify its specified vertices with distinct vertices from $X_i := \{c_i\} \cup \bigcup_{j \in [k] \setminus \{i\}} \{v_{i,j}\}$ (there are k vertices in this set). Let us call this graph I_i ('I' stands for *indicator*).

The construction forces that in every list homomorphism $f \colon I_i \to H$, at least one vertex from X_i is mapped to β. Observe that:

- for every $f \colon I_i \to H$, if $f(x_i) = v_i$, then $f(c_i) = \beta$.
- for every $j \neq i$, there exist $f', f'' \colon I_i \to H$, such that $f'(x_i) = f''(x_i) = v_j$ and $f'(c_i) = \alpha$ and $f''(c_i) = \beta$.

Step 2. Constructing a half-edge gadget. Let us construct k indicator gadgets $I_1, I_2, \ldots, I_k$. We identify the vertices $x_1, x_2, \ldots, x_k$, and call this vertex x. Call the resulting gadget a *half-edge*. By the construction of indicators, we observe that for a half-edge F, the following hold:

- for every $f \colon F \to H$, if $f(x) = v_i$, then $f(c_i) = \beta$,
- for every $i \in [k]$, and every tuple $X \in \{\alpha, \beta\}^k$, such that $X_i = \beta$, there exists $f \colon F \to H$ such that $f(x) = v_i$ and $f(c_j) = X_j$.

Step 3. Constructing an edge gadget. An *edge gadget* consists of two half-edge gadgets F, F' (we will use primes to denote the vertices in F'). Moreover, for every $i \in [k]$, we introduce a $NAND_2$ gadget on vertices c_i and c'_i, which enforces that at least one of them is mapped to α. We call the resulting graph an *edge gadget*. For an edge gadget FF we observe the following:

- for any $f \colon FF \to H$, if $f(x) = v_i$, then $f(x') \neq v_i$. Assume for contradiction that $f(x) = f(x') = v_i$. Then by the construction of a half-edge, we observe that $f(c_i) = f(c'_i) = \beta$. However, this is impossible by the definition of $NAND_2$ gadget.
- for any distinct $i, j \in [k]$ there is $g \colon FF \to H$ such that $g(x) = v_i$, and $g(x') = v_j$. By the construction of a half-edge gadget, there is $f \colon F \to H$, such that $f(x) = v_i$, $f(c_i) = \beta$, and $f(c_{i'}) = \alpha$ for every $i' \neq i$ (in particular, for $i' = j$). Analogously, there is $f' \colon F' \to H$, such that $f'(x') = v_j$, $f'(c_j) = \beta$ and $f'(c_{j'}) = \alpha$ for every $j' \neq j$. We obtain g by combining f and f', and extending this partial homomorphism to vertices of $NAND_2$-gadgets. By the definition of these gadgets, it is possible, as for every i' we have $f(c_{i'}) = \alpha$ or $f'(c_{i'}) = \alpha$.

Observe that the construction so far was performed for H only. Let $g(H)$ be the number of vertices in an edge gadget.

Step 4. Constructing G^* and H-lists L. We start constructing G^* by including the vertex set $V(G)$ of G to G^*(initially they are isolated vertices). For every edge uv of G, we introduce an edge gadget, where x is unified with u and x' is unified with v. By the construction of edge gadget, we observe that for every $v \in V(G)$ we have $L(v) = \mathcal{S}$. Moreover, (G^*, L) is a Yes-instance of LHom(H) if and only if G is k-colorable (we interpret mapping u to $v_i \in \mathcal{S}$ as coloring u with color i). Recall that the size of each edge gadget is $g(H)$, thus the number of vertices of G^* is at most $(|V(G)| + |E(G)|) \cdot g(H)$.

To see that the treewidth of G^* is at most $\text{tw}(G) + g(H)$, consider a tree decomposition $\mathcal{T}$ of G with width $\text{tw}(G)$. For every edge uv of G, we choose one bag X_{uv} of $\mathcal{T}$, such that $u, v \in X_{uv}$. Define a set X'_{uv} as the union of X_{uv} and the set of vertices of the edge gadget corresponding to the edge uv. We extend $\mathcal{T}$ to a tree decomposition $\mathcal{T}^*$ of G^*, by introducing a bag X'_{uv} for every edge uv of G and making it adjacent (in $\mathcal{T}^*$) to X_{uv} only. It is straightforward to verify that $\mathcal{T}^*$ is a tree decomposition of G^* of width at most $\text{tw}(G)+g(H)$. Moreover, it is clear that G^* and L can be constructed in time $g'(H) \cdot poly(|V(G)|)$ for some function g'. Thus, by Theorem 1, our claim holds. ◀

References

1 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74. ACM, 2007. `doi:10.1145/1250790.1250801`.

2 Hans L. Bodlaender, Paul S. Bonsma, and Daniel Lokshtanov. The fine details of fast dynamic programming over tree decompositions. In Gregory Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 41–53. Springer, 2013. `doi:10.1007/978-3-319-03898-8_5`.

3 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. `doi:10.1093/comjnl/bxm037`.

4 Andrei A. Bulatov. H-coloring dichotomy revisited. *Theor. Comput. Sci.*, 349(1):31–39, 2005. `doi:10.1016/j.tcs.2005.09.028`.

5 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 189–200. Springer, 2014. `doi:10.1007/978-3-662-44465-8_17`.

6 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. `doi:10.1109/FOCS.2011.23`.

7 Víctor Dalmau, László Egri, Pavol Hell, Benoit Larose, and Arash Rafiey. Descriptive complexity of list h-coloring problems in logspace: A refined dichotomy. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 487–498. IEEE Computer Society, 2015. `doi:10.1109/LICS.2015.52`.

8 László Egri, Pavol Hell, Benoit Larose, and Arash Rafiey. Space complexity of list H-colouring: a dichotomy. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth*

Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, pages 349–365. SIAM, 2014. doi:10.1137/1.9781611973402.26.

9 Tomás Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Comb. Theory, Ser. B*, 72(2):236–250, 1998. doi:10.1006/jctb.1997.1812.

10 Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999. doi:10.1007/s004939970003.

11 Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.

12 Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms of graphs with bounded degrees. *Discrete Mathematics*, 307(3-5):386–392, 2007. doi:10.1016/j.disc.2005.09.030.

13 Tomás Feder, Pavol Hell, Jing Huang, and Arash Rafiey. Interval graphs, adjusted interval digraphs, and reflexive list homomorphisms. *Discrete Applied Mathematics*, 160(6):697–707, 2012. doi:10.1016/j.dam.2011.04.016.

14 Tomás Feder, Pavol Hell, Sulamita Klein, and Rajeev Motwani. List partitions. *SIAM J. Discrete Math.*, 16(3):449–478, 2003. URL: http://epubs.siam.org/sam-bin/dbq/article/38405.

15 Tomás Feder, Pavol Hell, David G. Schell, and Juraj Stacho. Dichotomy for tree-structured trigraph list homomorphism problems. *Discrete Applied Mathematics*, 159(12):1217–1224, 2011. doi:10.1016/j.dam.2011.04.005.

16 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 142–151. SIAM, 2014. doi:10.1137/1.9781611973402.10.

17 Pavol Hell and Jaroslav Nesetril. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

18 Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. *Eur. J. Comb.*, 52:338–367, 2016. doi:10.1016/j.ejc.2015.07.011.

19 C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. URL: http://eudml.org/doc/213681.

20 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 777–789. SIAM, 2011. doi:10.1137/1.9781611973082.61.

21 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 760–776. SIAM, 2011. doi:10.1137/1.9781611973082.60.

22 Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

23 Mark H. Siggers. A New Proof of the H-Coloring Dichotomy. *SIAM Journal on Discrete Mathematics*, 23(4):2204–2210, 2010. doi:10.1137/080736697.

24 Zsolt Tuza. Graph colorings with local constraints - a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228, 1997. doi:10.7151/dmgt.1049.

25 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. `doi:10.1007/978-3-642-04128-0_51`.