The Complexity of Homomorphism Reconstruction Revisited

Timo Gervens
RWTH Aachen University, Germany

Martin Grohe
RWTH Aachen University, Germany

Louis Härtel
RWTH Aachen University, Germany

Philipp da Silva Fonseca
RWTH Aachen University, Germany

Abstract

We revisit the algorithmic problem of reconstructing a graph from homomorphism counts that has first been studied in (Böker et al., STACS 2024): given graphs $F_1, \dots, F_k$ and counts $m_1, \dots, m_k$, decide if there is a graph G such that the number of homomorphisms from F_i to G is m_i, for all i. We prove that the problem is NEXP-hard if the counts m_i are specified in binary and Σ_2^p-complete if they are in unary.

Furthermore, as a positive result, we show that the unary version can be solved in polynomial time if the constraint graphs are stars of bounded size.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Graph theory

Keywords and phrases graph homomorphism, nexp-complete, counting complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.45

Related Version *Full Version*: `https://arxiv.org/abs/2602.12780`

Funding *Timo Gervens, Martin Grohe, Louis Härtel*: European Union (ERC, SymSim, 101054974).

1 Introduction

Homomorphism counts reveal valuable information about graphs. A well-known theorem, due to Lovász [26], states that a graph G can be characterised up to isomorphism by the homomorphism counts $\hom(F, G)$ from all graphs F into G. In recent years, it has become increasingly clear that many natural properties of graphs are characterised by homomorphism counts from restricted graph classes [10, 11, 14, 16, 21, 27, 33, 34]. For example, the homomorphism counts from all cycles characterise the spectrum of a graph. Homomorphism counts from trees characterise a graph up to fractional isomorphism [11, 36], and homomorphism counts from planar graphs characterise a graph up to quantum isomorphism [27]. It has been suggested in [15] that homomorphism counts can be used to define vector embeddings of graphs in a systematic and principled way: for every class $\mathcal{F}$ of graphs we define a mapping $G \mapsto \big(\hom(F, G)\big)_{F \in \mathcal{F}} \in \mathbb{R}^{\mathcal{F}}$ mapping each graph G into the (potentially infinite dimensional) vector space $\mathbb{R}^{\mathcal{F}}$, the *latent space* of the embedding. We call such embeddings *homomorphism embeddings*. Vector embeddings of graphs are mainly of interest in a machine-learning context, because typical ML algorithms operate on vector representations of the data. Homomorphism embeddings have been shown to be useful in practice [20, 38], and they are related to other vector embeddings of graphs, such as those obtained from graph kernels or computed by graph neural networks (see [15]).

Now suppose we have a vector embedding and carry out computations in the latent space. For example, we may run an optimisation algorithm and find a point in the latent space that is optimal in some sense. How do we get back a graph from this point? For homomorphism embeddings, this is the question of how we reconstruct a graph from homomorphism counts. This is the central problem we study in this paper; let us state it formally.

> HOMREC
> **Input:** Pairs $(F_1, m_1), \ldots, (F_k, m_k)$, where $F_1, \ldots, F_k$ are graphs and $m_1, \ldots, m_k \in \mathbb{N}$ (in binary encoding).
> **Question:** Is there a graph G such that $\hom(F_i, G) = m_i$ for every $i \in [k]$?

Note that here we are looking at an embedding into a finite-dimensional latent space; the class $\mathcal{F}$ from the discussion above is $\{F_1, \ldots, F_k\}$ here.We call the pairs (F_i, m_i) *constraints*, the graphs F_i *constraint graphs*, and the numbers m_i the *counts*. This problem was first studied systematically by Böker, Härtel, Runde, Seppelt, and Standke [4]. The main results of that paper were various hardness and a few tractability results for restricted versions of the problem. For example, the problem is NP-hard even for constraint graphs of bounded tree width. The general problem was shown to be in NEXP and hard for the complexity class $\mathsf{NP}^{\#\mathsf{P}}$. The exact complexity of the problem was left open. Our first theorem settles this question.

▶ **Theorem 1.** HOMREC *is* NEXP*-complete.*

The proof of this result is a reduction from the SUCCINCTCLIQUE problem, known to be NEXP-complete from [31].

So far, following [4], we have always assumed the counts m_i in the constraints to be encoded in binary. However, it may be more natural to encode the numbers in unary. Let us call the resulting version of the problem UNHOMREC. The reason that UNHOMREC may be the more natural problem is the observation [4, Lemma 6] that if an instance $(F_1, m_1), \ldots, (F_k, m_k)$ has a solution satisfying all constraints, then it has a solution of order at most $\sum_{i=1}^{k} m_i |F_i|$, simply because we can delete all vertices not in the image of the homomorphisms required to satisfy the constraints. If the m_i are encoded in unary, this is polynomial in the input size. This puts UNHOMREC directly into the complexity class $\mathsf{NP}^{\#\mathsf{P}}$: we guess a solution and then verify that all constraints are satisfied using a #P-oracle. The exact complexity of UNHOMREC is lower, though.

▶ **Theorem 2.** UNHOMREC *is* Σ_2^p*-complete.*

So interestingly, by switching to a succinct encoding, we move the problem from Σ_2^p (rather than NP) to NEXP, which is a bit unusual: typically, succinct encodings of problems cause an exponential jump in complexity, from NP to NEXP [31], but here we only jump from Σ_2^p to NEXP.

Having now settled the exact complexity of the homomorphism reconstruction problem, both in its unary and binary versions, we set out to look for tractable special cases. For the binary version, Böker et al. [4] proved that reconstruction can be NP-hard even for a fixed set of constraint graphs that are all labelled trees. For the unary version, it is not clear if this can happen as well. But even for very simple special cases, it is not at all obvious how we can solve the reconstruction problem efficiently. Consider the case with two constraints $(\bullet, n)$, (, m), where the constraint graphs are a single vertex graph and the path of length 2. Intuitively, this is the problem of deciding if there is a graph G with n vertices and m homomorphic images of a path of length 2 in time polynomial in n and m. Arguably, the subgraph version of this problem is more natural: given n and m, decide if there is a graph with n vertices and m paths of length 2 as subgraphs. It is not obvious how to construct such a graph in time polynomial in n and m. (This is the simplest case that was left open in [4].) The following theorem implies that there is a polynomial-time algorithm.

Recall that a *star* is a tree of height 1, that is, a connected graph that has at most one vertex of degree greater than 1. Both the 1-vertex graph and the path of length 2 are examples of stars.

▶ **Theorem 3.** *The following problem is solvable in time* $m^{O(\ell^2)}$*, where* $\ell := \max\{|F_i| \mid i \in [k]\}$ *and* $m = \max\{m_i \mid i \in [k]\}$*.*

StarHomRec
Input: $(F_1, m_1), \dots, (F_k, m_k)$, where $F_1, \dots, F_k$ are stars and $m_1, \dots, m_k \in \mathbb{N}$ (in unary encoding)
Question: Is there a graph G such that $\hom(F_i, G) = m_i$ for all $i \in [k]$?

To prove Theorem 3, we observe that the homomorphism counts $\hom(S, G)$ from stars S into a graph G only depend on the degree sequence of G. Based on this observation, we devise a dynamic-programming algorithm that computes a feasible degree sequence and then reconstructs a graph from this degree sequence using a known algorithm due to Havel [19] and Hakimi [17].

Noting that the homomorphism counts from all stars of order at most ℓ determine the subgraph counts for all stars of order at most ℓ (see [9, 25]), as a corollary, we obtain the corresponding result for subgraph counts. In fact, it will be easier to prove the version for subgraph counts first and then derive the result for homomorphism counts.

▶ **Corollary 4.** *The following problem is solvable in time* $m^{O(\ell^2)}$*, where* $\ell := \max_{i \in [k]} |F_i|$ *and* $m = \max_{i \in [k]} m_i$*.*

StarSubRec
Input: $(F_1, m_1), \dots, (F_k, m_k)$, where $F_1, \dots, F_k$ are stars and $m_1, \dots, m_k \in \mathbb{N}$ (in unary encoding)
Question: Is there a graph G that has m_i subgraphs isomorphic to F_i, for every $i \in [k]$?

We note that for both Theorem 3 and Corollary 4, we can not only solve the decision problem, but actually construct a graph satisfying the constraints if there exists one.

Related Work

Homomorphisms and also homomorphism counts play an important role in many areas of mathematics and computer science and have also been studied in complexity theory. Leading to this paper is the recent work on homomorphism indistinguishability already discussed at the beginning of the introduction. Let us also mention work on the complexity of homomorphism reconstruction, from which we borrow some ideas [3, 6, 33].

The question of whether we can reconstruct a structure from substructures or substructure counts has a long tradition. Famously, the Ulam-Kelly Reconstruction Conjecture says that a graph of order n is uniquely determined by the multiset of its subgraphs of order $n - 1$ [22, 37] (also see [2, 30]). While the reconstruction conjecture is not algorithmic, there has been some work on the complexity of reconstructing graphs from small patterns [23].

Closely related to homomorphism reconstruction is the question of whether a graph can realise certain homomorphism densities, which is studied in extremal graph theory (e.g. [32]). An interesting undecidability result for reconstruction from homomorphism densities has been proved in [18].

In database theory, homomorphisms have played an important role since Chandra and Merlin's [7] characterisation of the containment problem for conjunctive queries in terms of homomorphisms. In a similar way, homomorphism counts are related to the containment problem for conjunctive queries under bag semantics [8]. Indeed, this problem is equivalent to the problem of deciding if, for two relational structures A, B, there is a structure X such that $\hom(A, X) > \hom(B, X)$. It is a long-standing open problem whether this is decidable (see [24, 28, 29] for partial results).

Organisation of the Paper

We prove Theorem 1 in Section 3, Theorem 2 in Section 4, and Theorem 3 in Section 5. Due to space limitations, many details will be deferred to the full version of the paper.

2 Preliminaries

We write $\mathbb{N} = \{0, 1, 2, \dots\}$ for the set of natural numbers, and we let $[k, \ell] := \{k, \dots, \ell\} \subseteq \mathbb{N}$ and $[k] := [1, k]$. A $\leq_p$ B denotes that a decision problem A is polynomial-time many-one reducible to the decision problem B. We denote the vertex set of a *graph* or directed acyclic graph (*dag*) G by $V(G)$ and the edge set by $E(G)$. For ease of notation, we denote an edge by uv or vu. A *homomorphism* from a graph F to a graph G is a mapping $h\colon V(F) \to V(G)$ such that $h(uv) \in E(G)$ for every $uv \in E(F)$. A *($\mathcal{C}$-vertex-)coloured graph* is a triple $G = (V, E, c)$ where (V, E) is a graph, the *underlying graph*, and $c\colon V(G) \to \mathcal{C}$ a function assigning a *colour* from a set $\mathcal{C}$ to every vertex of G. For $\mathsf{C} \in \mathcal{C}$, we let $\mathsf{C}^G := c^{-1}(\mathsf{C})$ be the *colour class* of C in G. An *($\mathcal{L}$-)labelled* graph is defined analogously with a function $\ell\colon \mathcal{L} \to V(G)$ assigning a vertex of G to every *label* from a set of labels $\mathcal{L}$ instead. Homomorphisms between coloured graphs and between labelled graphs are then defined as homomorphisms of the underlying graphs that respect colours and labels, respectively.

A graph G' is a *subgraph* of a graph G, written $G' \subseteq G$, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The *subgraph induced by a set* $U \subseteq V(G)$, written $G[U]$, is the subgraph of G with vertices U and edges $E(G) \cap U^2$. We write $\hom(F, G)$ for the number of homomorphisms from F to G and $\mathrm{sub}(F, G)$ for the number of subgraphs $G' \subseteq G$ such that $G' \cong F$. This notation generalises to coloured and labelled graphs in the straightforward way. For a vertex v in an undirected graph G, we let $N(v) := \{w \in V(G) \mid vw \in E(G)\}$ be the *neighbourhood* of v, and we let $\deg(v) := |N(v)|$ be the *degree* of v. For a vertex v in a dag G, we let $N^+(v) := \{w \in V(G) \mid vw \in E(G)\}$ be the *out-neighbourhood* of v and $N^-(v) := \{w \in V(G) \mid wv \in E(G)\}$ the *in-neighbourhood* of v; we let $\deg^+(v) := |N^+(v)|$ be the *out-degree* and $\deg^-(v) := |N^-(v)|$ the *in-degree*. We call nodes of in-degree 0 *sources* and nodes of out-degree 0 *sinks*.

The *degree sequence* of a graph is the sequence of its vertex degrees, ordered decreasingly. A sequence $\vec{d} = (d_1, \dots, d_n)$ of nonnegative integers I *graphic* if there is a graph G with degree sequence $\vec{d}$. For example, the sequence $(3, 3, 2, 2, 0)$ is graphic and the sequence $(3, 3, 1)$ is not. Note that if there is a graph G with degree sequence $(d_1, \dots, d_n)$ then $|V(G)| = n$ and $|E(G)| = \frac{1}{2}\sum_{i=1}^n d_i$.

A *boolean circuit* C is a dag where nodes are coloured with $\{\mathsf{In}, \vee, \wedge, \neg\}$. Nodes coloured $\vee$ or $\wedge$ have in-degree 2, the $\neg$ nodes in-degree 1, and we call them *gates*. In nodes $x_1, \dots, x_n$ are sources that we call *inputs*, and sinks $y_1, \dots, y_m$ *outputs*. A circuit C computes a function $C : \{0, 1\}^n \to \{0, 1\}^m$. The size of C, denoted by $|C|$, is the number of nodes in it.

3 NEXP-Completeness of Binary Homomorphism Reconstruction

In this section we prove Theorem 1. Whenever we use coloured graphs, denoted F or G, we will not explicitly state their colouring function $c : V(F) \to \mathcal{C}$. The set of colours $\mathcal{C}$ contains symbols such as P, α, or $\perp$, and abbreviations that begin with a capital letter such as C_i or B_i, whenever we need to parametrise colour classes with an index i.

Recall that the *constraints* of a HomRec instance are pairs (F, n) where F is a graph and n a nonnegative integer in binary representation. We denote the concatenation of lists of constraints as union, and we make no distinction between individual constraints and lists of length one. Note that the order in which the constraints appear in a HomRec-instance does not matter.

The missing proofs of Claim 7, Claim 10, and Claim 13 can be found in the full version.

3.1 Transforming Boolean Circuits into Homomorphism Constraints

As a warm-up result for proving that HomRec is NEXP-complete, we will show that ColHomRec, where the constraint graphs F_i (and the target graph G) may be coloured graphs, is NP-hard. While this is known [4], our new proof provides us with a good opportunity to introduce the main technique of Section 3.2 – representing boolean circuits in the graph we are aiming to reconstruct.

> CircuitSat
> **Input:** Circuit $C : \{0,1\}^n \to \{0,1\}$ where $n \in \mathbb{N}$.
> **Question:** Is there an input $x \in \{0,1\}^n$ such that $C(x) = 1$?

The coloured graph $G = (V(G), E(G), c : V(G) \to \mathcal{C})$ we will reconstruct, if an input x with $C(x) = 1$ exists, should contain an explicit representation of the computation of C on x. Gates will correspond to nodes, truth values will be represented as unique vertices of colours $\perp$ and $\top$, and edges from nodes encode a truth value depending on whether they connect them with $\perp$ or $\top$.

▶ **Theorem 5.** CircuitSat $\leq_p$ ColHomRec.

The $n : m$ constraint. In the construction of homomorphism constraints for representing the evaluation of boolean circuits, we will make use of constraints that enforce regularity between certain colour classes. By calculating appropriate multiplicities of their occurrences, we can let homomorphism counts from stars – rather, sums of powers of degrees – do the rest. For any two colours $\mathsf{A}, \mathsf{B} \in \mathcal{C}$ and positive integers $n, m \in \mathbb{N}$, we define the following list of constraints:

$$\mathcal{F}_{\equiv}(\mathsf{A}, \mathsf{B}, n, m) := \big((\bullet\mathsf{A}, n), (\mathsf{A}\bullet\!\!-\!\!\bullet\mathsf{B}, nm), (\overset{\mathsf{B}}{\bullet}\!-\!\overset{\mathsf{A}}{\bullet}\!-\!\overset{\mathsf{B}}{\bullet}, nm^2)\big).$$

▶ **Lemma 6.** *Any graph G that satisfies $\mathcal{F}_{\equiv}(\mathsf{A}, \mathsf{B}, n, m)$ contains a colour class A^G of size n. Each vertex coloured A has exactly m neighbours coloured B.*

Proof. Let G be some graph satisfying the constraints. Then its colour class A^G has size $|\mathsf{A}^G| = n$. Denote $\deg_{\mathsf{B}}(v) := |N(v) \cap \mathsf{B}^G|$. Observe that for the number of homomorphisms to G holds $\hom(\mathsf{A}\bullet\!\!-\!\!\bullet\mathsf{B}, G) = \sum_{v \in \mathsf{A}^G} \deg_{\mathsf{B}}(v) = nm$. Furthermore, we have $\hom(\overset{\mathsf{B}}{\bullet}\!-\!\overset{\mathsf{A}}{\bullet}\!-\!\overset{\mathsf{B}}{\bullet}, G) = \sum_{v \in \mathsf{A}^G} \deg_{\mathsf{B}}(v)^2 = nm^2$. By applying Cauchy–Schwarz it follows:

$$nm^2 = \sum_{v \in \mathsf{A}^G} \deg_{\mathsf{B}}(v)^2 \geq \frac{1}{n}\Big(\sum_{v \in \mathsf{A}^G} \deg_{\mathsf{B}}(v)\Big)^2 = \frac{1}{n}(nm)^2 = nm^2.$$

Equality in the Cauchy–Schwarz inequality holds if and only if for $v \in \mathsf{A}^G$ all $\deg_{\mathsf{B}}(v)$ are equal, therefore every vertex of colour A has exactly m neighbours in B. ◀

In order to create graphs that mirror the combinatorial structure of boolean circuits, we have to come up with a technique to represent different types of gates and logical truth values. We can greatly simplify this problem by imposing an order on the nodes in $V(C)$ and colouring them individually. This way, we can introduce homomorphism constraints that ensure that, in all occurrences of a reconstructed copy of the circuit C, locally the computation at each gate is correct. Truth values are represented as nodes being adjacent to either one of two vertices in the *value gadget*. For this gadget, which is unique and globally controls the evaluation at each gate, we reserve the colours $\{\alpha, \bot, \top\}$.

Let the *value gadget* G_α be the coloured graph $\overset{\bot\ \alpha\ \alpha\ \top}{\bullet\!-\!\bullet\!-\!\bullet\!-\!\bullet}$, and let

$$\mathcal{F}_\alpha := \big((\bullet\alpha, 2), (\bullet\bot, 1), (\bullet\top, 1), (\overset{\bot\ \alpha\ \alpha\ \top}{\bullet\!-\!\bullet\!-\!\bullet\!-\!\bullet}, 1), (\alpha\bullet\!-\!\bullet\bot, 1), (\alpha\bullet\!-\!\bullet\top, 1)\big).$$

Furthermore, for every colour A, let

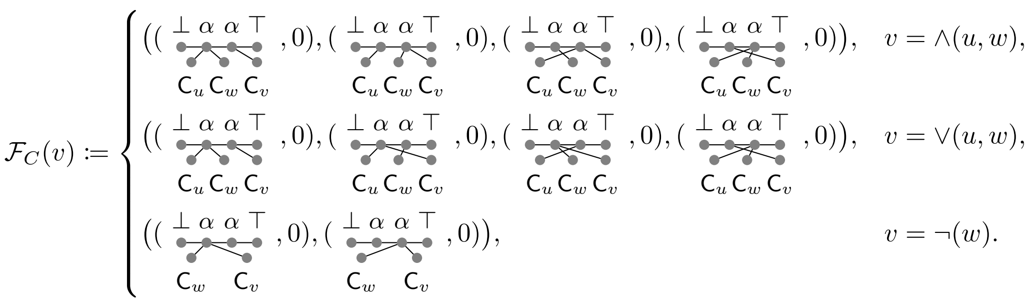
$$\mathcal{F}_{[\![\cdot]\!]}(\mathsf{A}, n) := \mathcal{F}_{\equiv}(\mathsf{A}, \alpha, n, 1).$$

If G satisfies the constraints $\mathcal{F}_\alpha \cup \mathcal{F}_{[\![\cdot]\!]}(\mathsf{A}, k)$ for some $k \in \mathbb{N}$ and $\mathsf{A} \in \mathcal{C}$, we can define the value function $[\![v]\!]_{\mathsf{A}} : \mathsf{A}^G \to \{0,1\}$ by letting $[\![v]\!]_{\mathsf{A}} = 1$, if v is adjacent to the α vertex next to $\top$, and $[\![v]\!]_{\mathsf{A}} = 0$, otherwise. We will simply write $[\![v]\!]$, if G is clear from context and the colour of v irrelevant.

Proof of Theorem 5. Let $C = (V(C), E(C), \mathsf{In}^C, \vee^C, \wedge^C, \neg^C)$ be a circuit with n input gates $v_1, \dots, v_n$ and 1 output gate o.

The Graph $G_{C(x)}$. For input $x \in \{0,1\}^n$, we define an undirected coloured graph $G_{C(x)}$ as follows: We start with a value gadget. Then for every vertex $v \in V(C)$, we introduce a vertex with colour C_v. Depending on the value of v in $C(x)$, the vertex v of $G_{C(x)}$ is connected to the α-vertex next to $\bot$ if the value is 0 or to the α-vertex next to $\top$ if the value is 1.

The Homomorphism Constraints. For gate $v \in \vee^C$ we write $v = \vee(u, w)$, if v has incoming edges $uv, wv \in E(C)$. Similarly, we write $v = \wedge(u, w)$ if $v \in \wedge^C$, or $\neg(w)$ if $v \in \neg^C$, for the other two types of gates. We define the constraints $\mathcal{F}_C(v)$ as follows:



We construct the following list of constraints $\mathcal{F}$ such that $\exists x \in \{0,1\}^n : C(x) = 1 \Leftrightarrow \mathcal{F} \in$ HomRec:

$$\mathcal{F} := \mathcal{F}_\alpha \cup \big((\overset{\mathsf{C}_o\ \alpha\ \top}{\bullet\!-\!\bullet\!-\!\bullet}, 1)\big) \cup \bigcup_{v \in V(C)} \mathcal{F}_{[\![\cdot]\!]}(\mathsf{C}_v, 1) \cup \bigcup_{v \in V(C) \setminus \mathsf{In}^C} \mathcal{F}_C(v).$$

If some $x \in \{0,1\}^n$ with $C(x) = 1$ exists, we observe that $G_{C(x)}$ as defined before satisfies all constraints in $\mathcal{F}$.

▷ Claim 7. If there is a graph G satisfying all constraints in $\mathcal{F}$, there exists an $x \in \{0,1\}^n$ such that $C(x) = 1$. Furthermore, $\mathcal{F}$ can be computed in polynomial time.

This completes the proof of Theorem 5. ◀

3.2 NEXP-Completeness of Coloured Homomorphism Reconstruction

NP-complete problems with exponentially succinct representations are often complete for nondeterministic exponential time NEXP [31]. The notion of succinct input representations was first investigated by Galperin and Wigderson in [13]. For example, a graph can be succinctly encoded by a boolean circuit as follows. We denote the binary encoding of a nonnegative integer $i \in \mathbb{N}$ by $\bar{i}$.

▶ **Definition 8** ([13]). *Let G be a graph with $m \leq 2^n$ vertices $v_0, \ldots, v_{m-1}$. A* succinct representation (SCR) *of a graph G is a boolean circuit C_G with $2n$ inputs and 1 output such that $C_G(\bar{i}, \bar{j}) = 1$ if $i, j < m$ and $(v_i, v_j) \in E(G)$ and $C_G(\bar{i}, \bar{j}) = 0$ otherwise.*

We reduce from the following SuccinctClique-problem, which is NEXP-complete [13].

SuccinctClique
Input: SCR C_G, positive integer $k \in \mathbb{N}$ (in binary).
Question: Does G contain a k-clique?

As the first and most difficult step in the proof of Theorem 1, the following lemma shows that ColHomRec is NEXP-hard.

▶ **Lemma 9.** SuccinctClique $\leq_p$ ColHomRec.

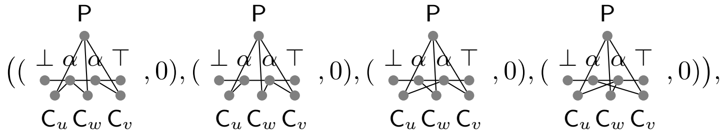
A Technical Change to the Encoding of Boolean Circuits from Section 3.1. We consider an instance (C_G, k) of SuccinctClique, where C_G is the SCR of a graph G. We let o be the output of C_G. Note that the circuit C_G has $2n$ inputs $v_0, \ldots, v_{2n-1}$ (rather than n as the circuits we considered before); we have to adapt our construction accordingly. Moreover, we will be creating $N := k(k-1)/2$ copies of the representation of the circuit C_G. For consistency, we keep only one value gadget. This means that our constraints need to change slightly.

We add constraints $\mathcal{F}_{\mathsf{P}}$ that will produce N vertices coloured P, one for each copy, which are connected to every vertex in their respective copies. The idea is that the P nodes together with their representation of C_G will form disjoint stars that can be evaluated independently from each other. We would prefer to individualize each C_G with its completely own set of colours $\mathsf{C}_v^{(1)}, \ldots, \mathsf{C}_v^{(k)}$ instead, but have to avoid introducing $\Omega(k)$ constraints. Let

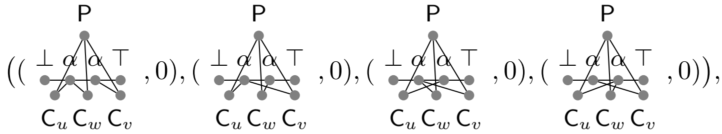
$$\mathcal{F}_{\mathsf{P}} := \bigcup_{v \in V(C_G)} \mathcal{F}_{\equiv}(\mathsf{P}, \mathsf{C}_v, N, 1) \cup \bigcup_{v \in V(C_G)} \mathcal{F}_{\equiv}(\mathsf{C}_v, \mathsf{P}, N, 1).$$

$\mathcal{F}_{\mathsf{P}}$ forces the colour classes P and C_v to have N vertices. Moreover, it forces each vertex in P to have 1 neighbour in C_v and each vertex in C_v to have 1 neighbour in P. The constraints $\mathcal{F}_{C_G}(v)$ enforcing correct behaviour of Boolean connectives are made less restrictive, by limiting them to $\mathsf{C}_u, \mathsf{C}_w$, and C_v nodes adjacent to the same P vertex:

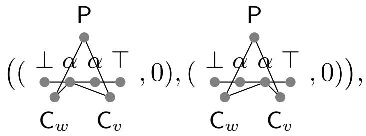
$\mathcal{F}_{C_G}(v) :=$



$v = \wedge(u, w)$,



$v = \vee(u, w)$,

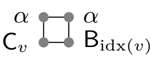
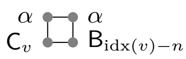


$v = \neg(w)$.

So far, we only guarantee the existence of $\binom{k}{2}$ copies of the SCR C_G in the graph we are trying to reconstruct. We need additional constraints $\mathcal{F}_{\mathsf{Q}}, \mathcal{F}_{\text{in}}$, and $\mathcal{F}_{\text{str}}$ to ensure that not only are there $\binom{k}{2}$ edges in G, but that k mutually adjacent vertices $S \subseteq V(G)$ forming a k-clique exist. The vertices in S will be represented as k node gadgets encoding their respective indices in G. The index of each node gadget (in binary representation) is fed into $k-1$ copies of C_G, which will have to be correctly evaluated by any graph satisfying our list of constraints. For node gadgets, we introduce the colours B_i, for $i \in [0, n-1]$, and Q. We denote the first and second set of n inputs to C_G as $\mathsf{In}_1 := \{v_0, \ldots, v_{n-1}\} \subseteq V(C_G)$ and $\mathsf{In}_2 := \{v_n, \ldots, v_{2n-1}\} \subseteq V(C_G)$, respectively, and as index $\text{idx}(v_i) := i$. Let

$$\mathcal{F}_{\mathsf{Q}} := \bigcup_{i\in[0,n-1]} \mathcal{F}_{[\![\cdot]\!]}(\mathsf{B}_i, k) \cup \bigcup_{i\in[0,n-1]} \mathcal{F}_{\equiv}(\mathsf{Q}, \mathsf{B}_i, k, 1) \cup \bigcup_{i\in[0,n-1]} \mathcal{F}_{\equiv}(\mathsf{B}_i, \mathsf{Q}, k, 1),$$

$$\mathcal{F}_{\text{in}} := \bigcup_{v\in\mathsf{In}_1} \mathcal{F}_{\equiv}(\mathsf{C}_v, \mathsf{B}_{\text{idx}(v)}, N, 1) \cup \bigcup_{v\in\mathsf{In}_2} \mathcal{F}_{\equiv}(\mathsf{C}_v, \mathsf{B}_{\text{idx}(v)-n}, N, 1)$$

$\cup \bigcup_{v\in\mathsf{In}_1}$ ( , 0) $\cup \bigcup_{v\in\mathsf{In}_2}$ ( , 0).

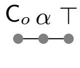
Combining $n : m$ Constraints Guarantees Consistent Circuit Inputs. The constraints in $\mathcal{F}_{\mathsf{Q}}$ enforce colours classes Q and B_i of size k. Again, vertices coloured Q are required to have 1 neighbour coloured B_i, and vice versa. Furthermore, B_i has to define the value function $[\![\cdot]\!]$.

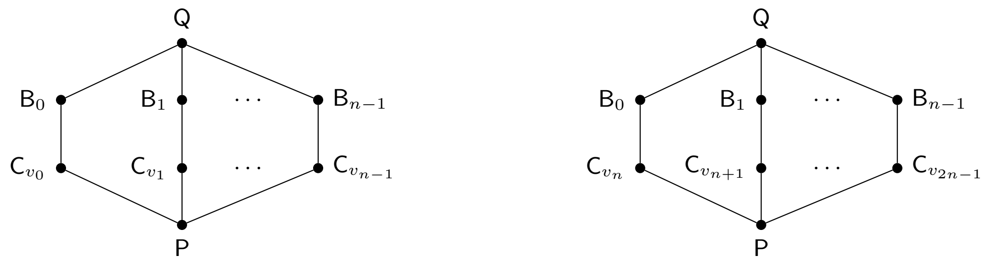
For inputs $v \in \mathsf{In}^{C_G}$, $\mathcal{F}_{\text{in}}$ enforces that each vertex coloured C_v has exactly one neighbour coloured $\mathsf{B}_{\text{idx(v)}}$ or $\mathsf{B}_{\text{idx(v)}-n}$, depending on whether $v \in \mathsf{In}_1$ or $v \in \mathsf{In}_2$ holds. Here, we do not require the converse, since we would like every vertex coloured B_i to have $k-1$ neighbours total in C_{v_i} and $\mathsf{C}_{v_{i+n}}$. The third row of constraints enforces the values $[\![\cdot]\!]$ of vertices in C_v and their neighbours in $\mathsf{B}_{\text{idx(v)}}$ or $\mathsf{B}_{\text{idx(v)}-n}$ to be equal.

To make the indices of node gadgets behave as input strings of length n, we introduce the two constraints shown in Figure 1, and let

$$\mathcal{F}_{\text{str}} := \big((F_{\mathsf{In}_1}, N), (F_{\mathsf{In}_2}, N)\big).$$

Proof of Lemma 9. We combine the previous constraints into $\mathcal{F}^+ := \mathcal{F}_{\mathsf{P}} \cup \mathcal{F}_{\mathsf{Q}} \cup \mathcal{F}_{\text{in}} \cup \mathcal{F}_{\text{str}}$. We will show that for any SCR C_G and $k \in \mathbb{N}$ holds that G contains mutually adjacent $S = \{s_1, \ldots, s_k\} \Leftrightarrow \mathcal{F} \in \textsc{HomRec}$, for the following list of constraints where $N := k(k-1)/2$:

$\mathcal{F} := \mathcal{F}_\alpha \cup \big(($  $, N)\big) \cup \bigcup_{v\in V(C_G)} \mathcal{F}_{[\![\cdot]\!]}(\mathsf{C}_v, N) \cup \bigcup_{v\in V(C_G)\setminus\mathsf{In}} \mathcal{F}_{C_G}(v) \cup \mathcal{F}^+.$



(a) The graph F_{In_1}. **(b)** The graph F_{In_2}.

Figure 1 These constraint graphs are the only configurations between inputs and vertices encoding indices we want to allow. Recall that the set of $2n$ inputs to C_G is $v_0, \ldots, v_{n-1}, v_n, \ldots, v_{2n-1}$.

The Graph G_S. How does a graph satisfying the constraints $\mathcal{F}$ look like? We have a good idea about the first part; it needs to contain $k(k-1)/2$ copies of the SCR C_G, which all evaluate to 1. But the second half of our constraints, $\mathcal{F}^+$, interacts with the inputs to our circuits C_G.

For input SCR C_G, we define an undirected coloured graph G_S for any size k subset of vertices $S \subseteq V(G)$ as follows:

- Let $S = \{s_1, \ldots, s_k\}$, and denote their binary encodings with $\overline{s_i} \in \{0,1\}^n$.
- Recall that G_α denotes the value gadget $\perp\ \alpha\ \alpha\ \top$. Denote its vertices, left to right, as $\{v_\perp, v_{\alpha\perp}, v_{\alpha\top}, v_\top\}$.
- Let $V_{\text{bit}} := \{b_0, \ldots, b_{n-1}\}$.
- We will be using $\binom{S}{2} = \{s_1s_2, \ldots, s_{k-1}s_k\}$ to index the set $[1, N] = [1, k(k-1)/2]$.
- The colours of G_S are $\mathcal{C} := \{\alpha, \top, \perp, \mathsf{P}, \mathsf{Q}\} \cup \{\mathsf{C}_v \mid v \in V(C_G)\} \cup \{\mathsf{B}_i \mid i \in [0, n-1]\}$.
- Let $V(G_S) := V(G_\alpha) \cup \left(\{v_\mathsf{P}\} \times \binom{S}{2}\right) \cup \left(V(C_G) \times \binom{S}{2}\right) \cup \left(\{v_\mathsf{Q}\} \times S\right) \cup \left(V_{\text{bit}} \times S\right)$.
- We let its colour classes (omitting G_S) $\alpha := \{v_{\alpha\perp}, v_{\alpha\top}\}$, $\perp := \{v_\perp\}$, and $\top := \{v_\top\}$;
- $P := \{v_\mathsf{P}\} \times \binom{S}{2}$, $Q := \{v_\mathsf{Q}\} \times S$;
- for $v \in V(C_G)$, $\mathsf{C}_v := \{v\} \times \binom{S}{2}$; and, for $i \in [0, n-1]$, $\mathsf{B}_i := \{v_i\} \times S$.
- We define its edges $E(G_S)$ as follows: any edges from $E(G_\alpha)$.
- For each $s_is_j \in \binom{S}{2}$, the vertex (v_P, s_is_j) coloured P has one neighbour (v, s_is_j) with colour C_v for each $v \in V(C_G)$.
- For each $s \in S$, the vertex (v_Q, s) coloured Q has one neighbour (b_i, s) with colour B_i for each $i \in [0, n-1]$.
- For each $i \in [0, n-1]$ and $s \in S$, the vertex (b_i, s) is adjacent to (v, s_is_j) if v is the i-th input of C_G and $s = s_i$ OR if v is the $n+i$-th input of C_G and $s = s_j$, and
- the vertex (b_i, s) is adjacent to $v_{\alpha\perp}$, if $\overline{s}_i = 0$, and adjacent to $v_{\alpha\top}$, if $\overline{s}_i = 1$.
- Observing that this defines $[\![\cdot]\!]$ on the inputs of all copies of C_G, we can define the remaining values of C_G copies inductively, by adding edges from gates to either $v_{\alpha\perp}$ or $v_{\alpha\top}$, as in the proof of Theorem 5.

If the graph G represented by SCR C_G contains some k-clique $S = \{s_1, \ldots, s_k\}$, then we argue that the graph G_S satisfies our constraints.

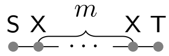
$\triangleright$ Claim 10. If there is a graph G satisfying all constraints in $\mathcal{F}$, there exist k strings $\overline{i_1}, \ldots, \overline{i_k} \in \{0,1\}^n$ such that pairwise $C_G(\overline{i}, \overline{j}) = 1$, and thus, there are k mutually adjacent vertices in the graph encoded by SCR C_G. Furthermore, $\mathcal{F}$ can be computed in polynomial time in $|C_G|$.

Claim 10 completes the proof of Lemma 9. $\blacktriangleleft$

3.3 Hardness for Uncoloured Graphs

To prove Theorem 1, we will show that for any instance of ColHomRec we can compute an equivalent instance of HomRec where all constraint graphs are simple uncoloured graphs in polynomial time.

▶ **Lemma 11.** *There exists a polynomial time reduction from* ColHomRec *(with an unbounded number of colour classes) to* ColHomRec *with at most* 4 *colour classes.*

Let the *m-colour gadget* G_m be the 3-colour graph 

, and let

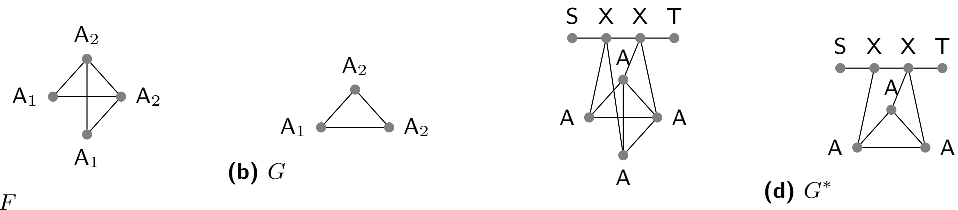
$$\mathcal{F}_X \coloneqq \big((\bullet\mathsf{X}, m), (\bullet\mathsf{S}, 1), (\bullet\mathsf{T}, 1), (\mathsf{X}\bullet\!\!-\!\!\bullet\mathsf{S}, 1), (\mathsf{X}\bullet\!\!-\!\!\bullet\mathsf{X}, 2m-2), (\mathsf{X}\bullet\!\!-\!\!\bullet\mathsf{T}, 1), (G_m, 1), \bigcup_{k<m} (G_k, 0)\big).$$

Observe that the m-colour gadget G_m satisfies $\mathcal{F}_\mathsf{X}$. Any other graph G that satisfies $\mathcal{F}_\mathsf{X}$ contains a single copy of G_m as subgraph. Furthermore, G cannot contain other vertices coloured S, T, or X.

Proof of Lemma 11. From an m-colour constraint graph $F = (V(F), E(F), \mathsf{A}_1^F, \dots, \mathsf{A}_m^F)$ we compute a 4-colour graph $F^* = (V(F^*), E(F^*), \mathsf{S}^{F^*}, \mathsf{T}^{F^*}, \mathsf{X}^{F^*}, \mathsf{A}^{F^*})$ as follows:

- $V(F^*) \coloneqq V(F) \cup \{s_S, s_1, \dots, s_m, s_T\}$,
- $E(F^*) \coloneqq E(F) \cup \bigcup_{i\in[m]} \{vs_i \mid v \in \mathsf{A}_i^F\} \cup \{s_i s_{i+1} \mid i \in [m-1]\} \cup \{s_S s_1, s_m s_T\}$,
- $\mathsf{S}^{F^*} \coloneqq \{s_S\}$, $\mathsf{T}^{F^*} \coloneqq \{s_T\}$, $\mathsf{X}^{F^*} \coloneqq \{s_i \mid i \in [m-1]\}$, and $\mathsf{A}^{F^*} \coloneqq V(F)$.

The graph F^* is the union of F and G_m, replacing any colour from F with colour A and an edge to G_m. Observe that the example shown in Figure 2 preserves the number of homomorphisms for each pair of graphs. In the following proof of Claim 12, we will make this fact explicit.

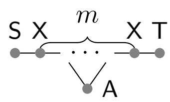


(a) F **(b)** G **(c)** F^* **(d)** G^*

Figure 2 F is a K_4 minus one edge; two vertices share colour A_1 so the missing edge is between them (this enables a homomorphism to G). G is the triangle $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_2$. F^* (resp. G^*) augments the base graph by a 2-*colour gadget* $\mathsf{S}-\mathsf{X}-\mathsf{X}-\mathsf{T}$ and, in correspondence to their previous colour, connects the base vertices to the gadget.

▷ Claim 12. For all m-colour graphs F, G and $n \in \mathbb{N}$ holds that G satisfies (F, n) if and only if G^* satisfies (F^*, n).

Proof of Claim 12. In any homomorphism $h : F^* \to G^*$, the m-colour gadget of F^* has to be mapped identically to the m-colour gadget of G^*. If F^* contains any A-coloured vertices, because they have exactly one X-neighbour, $vs_i \in E(F^*)$ holds if and only if $h(v)s_i \in E(G^*)$ does. By construction this is equivalent to $v \in \mathsf{A}_i^F \Leftrightarrow h(v) \in \mathsf{A}_i^G$, and thus, it holds that $\hom(F, G) = |\{h|_{V_F} \mid h \in \mathrm{Hom}(F^*, G^*)\}| = \hom(F^*, G^*)$. ◁

For all $i < j \in [m]$ we define the modified m-colour gadget $C_{m,i,j} :=$ 

,

where the A vertex is connected with the ith and jth X vertex.

Let $\mathcal{F} := (F_i, m_i)_{i \in [k]}$, where F_i are coloured with $\mathsf{A}_1, \dots, \mathsf{A}_m$. The reduction produces the following constraints $\mathcal{F}^*$ with 4 colour classes:

$$\mathcal{F}^* := \big((F_i^*, m_i)_{i \in [k]} \cup \mathcal{F}_X \cup (C_{m,i,j}, 0)_{i<j \in [m]}\big).$$

To show that $\mathcal{F} \in$ ColHomRec $\Leftrightarrow \mathcal{F}^* \in$ ColHomRec, first, assume that the m-colour graph $G = (V(G), E(G), \mathsf{A}_1^G, \dots, \mathsf{A}_m^G)$ satisfies $\mathcal{F}$.

Then $G^* = (V(G^*), E(G^*), \mathsf{S}^{G^*}, \mathsf{T}^{G^*}, \mathsf{X}^{G^*}, \mathsf{A}^{G^*})$, the graph we obtain from G with the same construction that produced the list of constraints, satisfies $(F_i^*, m_i)_{i \in [k]}$ because of Claim 12. Since G^* contains G_m as subgraph, and all other vertices are coloured A, G^* satisfies $\mathcal{F}_{\mathsf{X}}$. Furthermore, each vertex $v \in V(G^*)$ has at most one X-coloured neighbour, and thus, G^* satisfies $(C_{m,i,j}, 0)$ for each $i < j \in [m]$.

▷ Claim 13. If a 4-colour graph G satisfies $\mathcal{F}^*$, then there exists an m-colour graph G' that satisfies $\mathcal{F}$. Furthermore, $\mathcal{F}^*$ can be computed in polynomial time from $\mathcal{F}$.

Claim 13 completes the proof of Lemma 11. ◀

▶ **Lemma 14.** *There exists a polynomial time reduction from* ColHomRec *with at most* 4 *colour classes to* HomRec *(with uncoloured simple graphs).*

This part of the reduction requires a finite number of homomorphically incomparable graphs, called Kneser graphs, that behave similarly to colours when they are attached to vertices. We modify the constraints (F, m) by attaching indicator gadgets to all vertices of F. Because we do not disallow adjacency between vertices of the same colour, we cannot avoid replacing edges with the bidirectional gadget from [4]. For further details, we refer to the full version.

4 Σ_2^p-Completeness of Unary Homomorphism Reconstruction

In contrast to HomRec, the number of homomorphisms in constraints of UnHomRec is polynomially bound by the input size. As we will see, UnHomRec is contained in the second level of the polynomial hierarchy. In this section we will be talking about labelled graphs F or G, with a set of labels $\mathcal{L} := \{\ell_1, \dots, \ell_m\}$ for some $m \in \mathbb{N}$, and some labelling function $\ell : \mathcal{L} \to V(F)$ that we do not explicitly refer to.

▶ **Lemma 15.** UnHomRec $\in \Sigma_2^p$.

Proof. We can assume that the graph G has at most $\sum_{i=1}^{m} h_i \, |V(F_i)|$ vertices by the observation from [4]. An $\mathsf{NP}^{\mathsf{NP}}$ machine first non-deterministically guesses G and h_i different homomorphisms $h : F_i \to G$. We verify that there are no other homomorphisms by calling the NP oracle. ◀

▶ Remark 16. We remark that for constraint graphs from a class of bounded treewidth, the problem UnHomRec is in NP since counting homomorphisms is in polynomial time. It remains open, whether there are classes for which this problem is NP-complete.

The 2-Round-3-Colouring problem is complete for Σ_2^p (see Thm. 11.4 in [1]). Its containment in Σ_2^p is obvious, since it is of the form "$x \in L$ if and only if y exists such that $f(x, y) \in$ Non-3-Colouring" and we recall that Non-3-Colouring $\in$ coNP.

> 2-ROUND-3-COLOURING
> **Input:** Graph F.
> **Question:** Is there a 3-colouring of the degree 1 vertices of F that cannot be extended to a 3-colouring of F?

▶ **Lemma 17.** 2-ROUND-3-COLOURING $\leq_p$ UNHOMREC.

Proof. We will use labelled graphs as constraints. Encoding the labels as Kneser graph gadgets (see uncoloured hardness for $\mathsf{NP}^{\#\mathsf{P}}$ in [5], Appendix C), this reduction can be adapted to produce unlabelled graphs instead.

Let F be an instance of 2-ROUND-3-COLOURING, and fix an order $\{v_1, \ldots, v_m\}$ on all degree 1 vertices of F. We define the labelled graph F' by labelling each v_i by a distinct label ℓ_i. The following constraints are created by the reduction:

1. $\hom(F') = 0$,
2. $\hom(\bullet) = 3$,
3. $\hom(\bullet\!-\!\bullet) = 6$, and
4. $\hom(\bullet\ \ell_i) = 1$ for every $i \in [m]$.

The graph $\triangle$ is the unique graph that satisfies $\hom(\bullet) = 3$ and $\hom(\bullet\!-\!\bullet) = 6$, and each label from $\{\ell_1, \ldots, \ell_m\}$ has to appear exactly once in any graph G satisfying the constraints. If G satisfies $\hom(F') = 0$, then matching the position of labels in $\triangle$ to colours of labelled vertices in F' yields a 3-colouring of the degree 1 vertices of F that cannot be extended.

Conversely, assume that $\hom(F') > 0$ for all graphs G satisfying these constraints. Then each homomorphism from F' to G can be translated into a 3-colouring of F extending a 3-colouring of its degree 1 vertices, and since each 3-colouring of degree 1 vertices is represented by some position of labels in $\triangle$, it follows that $F \notin$ 2-ROUND-3-COLOURING. ◀

5 Tractability of Homomorphism Reconstruction for Star Counts

In this section we will derive an algorithm for STARHOMREC to prove Theorem 3 and thereby answer an open question from [4].

We will observe that the homomorphism counts $\hom(S, G)$ from stars S into a graph G only depend on the degree sequence of G. Exploiting this, our algorithm first reconstructs a degree sequence consistent with the counts, if it exists. Then it uses the Havel-Hakimi algorithm [17, 19] to construct a graph G with this degree sequence.

We use dynamic programming to reconstruct the degree sequence. For that we will derive a recursion of the following form: Given an instance $(F_1, m_1), \ldots, (F_k, m_k)$ of STARHOMREC, there exists a graph G with degree sequence $(d_1, \ldots, d_n)$ and $\hom(F_i, G) = m_i$ for $i \in [k]$ if and only if there exists a graph G' with degree sequence $(d'_1, \ldots, d'_{n'})$ and $\hom(F_i, G) = m'_i$ for $i \in [k]$, where

- the counts $m'_1, \ldots, m'_k$ only depend on $m_1, \ldots, m_k$ and d_1,
- the counts get smaller in each recursion step ($m'_i \leq m_i$ for $i \in [k]$),
- the graph gets smaller in each recursion step ($n < n'$),
- $(d_2, \ldots, d_n)$ can be computed from $(d'_1, \ldots, d'_{n'})$ and d_1 in polynomial time.

This allows us to recursively guess d_1 and compute the counts $m'_1, \ldots, m'_k$ from $m_1, \ldots, m_k$ and d_1. Then we get up to d_1 different but smaller instances of STARHOMREC that we have to compute. Since we store all computed results for STARHOMREC in a table, the number of recursive calls is still polynomial.

The exact recursion of our algorithm is slightly different (see Algorithm 1). This is, for example, because we must only consider graphic degree sequences. We address this challenge in Section 5.2, after showing a recursive formulation of $\hom(S, G)$ for stars S in Section 5.1. In Section 5.3 we present the final algorithm and prove Theorem 3.

5.1 Recursive Formulation of Star Counts

By S_j, we denote the star with j leaves. For any star S_j, graph G, and $v \in G$ we write $\hom(S_j, v)$ for the number of homomorphisms from S_j into G for which the star center node is mapped to v. We use $\mathrm{sub}(S_j, v)$ analogously for subgraphs.

▶ **Observation 18.** *For any graph G it holds that*

$$\begin{aligned}\hom(S_j, G) &= \sum_{v \in G} \hom(S_j, v) \quad \textit{for } j \in \mathbb{N},\\ \mathrm{sub}(S_j, G) &= \sum_{v \in G} \mathrm{sub}(S_j, v) \quad \textit{for } j \in \mathbb{N} \setminus \{1\},\\ 2 \cdot \mathrm{sub}(S_1, G) &= \sum_{v \in G} \mathrm{sub}(S_1, v).\end{aligned}$$

Note that $\mathrm{sub}(S_0, G) = \hom(S_0, G) = |V(G)|$ and $2 \cdot \mathrm{sub}(S_1, G) = \hom(S_1, G) = 2|E(G)|$.

The star counts for one vertex only depend on the vertex' degree.

▶ **Observation 19.** *For any graph G and vertex $v \in G$ with degree d it holds that*

$$\hom(S_j, v) = d^j, \qquad \mathrm{sub}(S_j, v) = \binom{d}{j}.$$

Overall, a graph's structure beyond its degree sequence is irrelevant for its star counts. To find the recursion for our algorithm we start with recursive formulations for the vertex based counts. For subgraphs we use

$$\mathrm{s}_j(d) := \begin{cases} 1 & \text{if } \ j = 0,\\ 0 & \text{if } \ d = 0, \ j > 0,\\ \mathrm{s}_j(d-1) + \mathrm{s}_{j-1}(d-1) & \text{otherwise.}\end{cases}$$

And for homomorphisms we use

$$\mathrm{h}_j(d) := \begin{cases} 1 & \text{if } \ j = 0,\\ 0 & \text{if } \ d = 0, \ j > 0,\\ \displaystyle\sum_{i=0}^{j} \binom{j}{i} \mathrm{h}_i(d-1) & \text{otherwise.}\end{cases}$$

▶ **Lemma 20.** *For any $j \in \mathbb{N}$, graph G, and vertex $v \in G$ with degree d it holds that*

$$\hom(S_j, v) = \mathrm{h}_j(d), \qquad \mathrm{sub}(S_j, v) = \mathrm{s}_j(d).$$

The proof of Lemma 20 can be found in the full version. In the next section, we will develop a recursion on graphic degree sequences for which we use above recursive formulation.

5.2 Recursion on Graphic Degree Sequences

Our reconstruction algorithm has to consider exactly all graphic degree sequences. In this section we explore a way to recursively shrink graphic degree sequences and recompute the subgraph counts for the shrunken sequence. The following theorem gives a criterion for a sequence of integers being graphic.

▶ **Theorem 21** (Erdős-Gallai [12]). *A non-increasing sequence $(d_1, \ldots, d_n)$ of non-negative integers is graphic if and only if $\sum_{i=1}^{n} d_i$ is even and for every k:*

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k). \tag{1}$$

Intuitively, for every selection of the k highest degree vertices, we need enough vertices to connect them to. To achieve this, the Havel-Hakimi algorithm [17, 19] (also see [35]) recursively sorts the sequence non-increasingly and then removes the first degree (d_1) while decrementing the next d_1 degrees. We use a similar (but slightly different) recursive operation.

▶ **Lemma 22.** *A non-increasing sequence $(d_1, \ldots, d_n)$ of positive integers with $n \geq 2$ and $d_1 \leq n-1$ is graphic if and only if the sequence $(d_2 - 1, \ldots, d_n - 1, \underbrace{1, \ldots, 1}_{n-1-d_1 \text{ times}})$, ordered decreasingly, is graphic.*

The proof of Lemma 22 can be found in the full version. We can relate the star counts of both sequences (before and after our operation) in Lemma 23 without knowing anything about the sequences besides d_1.

▶ **Lemma 23.** *Let $(d_1, \ldots, d_n)$ be a non-increasing sequence of positive integers with $n \geq 2$. Furthermore, let G be a graph with the degree sequence $(d_1, \ldots, d_n)$, and let G' be a graph with the degree sequence $(d_2 - 1, \ldots, d_n - 1, \underbrace{1, \ldots, 1}_{x})$ (ordered decreasingly) for any $x \in \mathbb{N}$. Then*

$$\begin{aligned}
\operatorname{sub}(S_0, G) &= \mathrm{s}_0(d_1) + \operatorname{sub}(S_0, G') - x,\\
2 \cdot \operatorname{sub}(S_1, G) &= \mathrm{s}_1(d_1) + 2 \cdot \operatorname{sub}(S_1, G') + \operatorname{sub}(S_0, G') - 2x,\\
\operatorname{sub}(S_2, G) &= \mathrm{s}_2(d_1) + \operatorname{sub}(S_2, G') + 2 \cdot \operatorname{sub}(S_1, G') - x,\\
\operatorname{sub}(S_j, G) &= \mathrm{s}_j(d_1) + \operatorname{sub}(S_j, G') + \operatorname{sub}(S_{j-1}, G') \quad \text{for } j \geq 3.
\end{aligned}$$

The proof of Lemma 23 can be found in the full version. It is based on the recursion from Section 5.1 and the factor 2 for $\operatorname{sub}(S_1, G)$ comes from Observation 18. An analogous version of Lemma 23 for homomorphisms could be made using the recursion for homomorphisms from Section 5.1 and leaving out the factor 2. However, for simplicity, we only consider subgraph counts in the remainder of Section 5. As observed in the introduction, for stars, subgraph counts determine homomorphism counts and vice versa, so this is sufficient.

It seems like this operation does not help us for our algorithm because the sequence gets longer. However, since the last x entries of the sequence all have value 1, we can separate this part from the (still unknown) degree sequence and carry the number x from one recursion step to another, updating it each step.

Intuitively, if we decrement more degrees than necessary to ensure that the degree d_1 can be realized, then, when another degree would not be realisable using only the rest of the degree sequence, we can use up x.

5.3 The Reconstruction Algorithm Using Dynamic Programming

In this section we first lift our recursion on graphic degree sequences to a recursion on the reconstruction problem. Then we present the reconstruction algorithm which is based on this recursion and discuss its runtime to finally prove Theorem 3.

▶ **Definition 24.** *We define the function* $\mathrm{DP} : \mathbb{N}^{\ell+3} \to \{0,1\}$ *with* $\mathrm{DP}(s_0, \dots, s_\ell, m, x) = 1$ *if there exist integers* $d_1 \geq \dots \geq d_{s_0}$ *and a graph* G *with degree sequence*

$$(d_1, \dots, d_{s_0}, \underbrace{1, \dots, 1}_{x\ \textit{times}}),$$

which fulfils

$$\begin{aligned} \mathrm{sub}(S_0, G) &= \mathrm{s}_0 + x, \\ 2 \cdot \mathrm{sub}(S_1, G) &= \mathrm{s}_1 + 2x, \\ \mathrm{sub}(S_2, G) &= \mathrm{s}_2 + x, \\ \mathrm{sub}(S_j, G) &= \mathrm{s}_j \qquad \textit{for } j \in [3, \ell], \\ d_i \leq m &\qquad \textit{for } i \in [1, s_0], \end{aligned}$$

and $\mathrm{DP}(s_0, \dots, s_\ell, m, x) = 0$ *otherwise.*

Note that $\mathrm{DP}(s_0, s_1/2, s_3, \dots, s_\ell, s_0 - 1, 0) = 1$ if and only if there exists a graph G with $\mathrm{sub}(S_j, G) = s_j$ for all $j \in [0, \ell]$. Intuitively, the parameter x corresponds to the x that we discussed in Section 5.2 and m acts as a maximum allowed degree, ensuring the (degree) sequence to be non-increasing which is a requirement for Lemma 22 and Lemma 23.

Using this definition we can solve the problem recursively up to the "base cases" with $s_0 < 2$ or $s_1 = 0$.

▶ **Lemma 25.** *For* $\ell \geq 1$ *and all non-negative integers* $s_0, \dots, s_\ell, m, x$ *with* $s_0 \geq 2$ *and* $s_1 \geq 1$ *it holds that* $\mathrm{DP}(s_0, \dots, s_\ell, m, x)$ *is equal to*

$$\begin{aligned} \max(\{\mathrm{DP}(s'_0, \dots, s'_\ell, m', x') \mid \exists d \in [1, m],\ \exists n_1 \in [1, s_0] :\ & \\ & d \leq n_1 - 1 + x, \\ & s'_0 = n_1 - 1, \\ & s'_j = s_j - s_j(d) - s'_{j-1} \quad \textit{for } j \in [1, \ell], \\ & m' = d - 1, \\ & x' = x + n_1 - 1 - d\}). \end{aligned}$$

In the proof of Lemma 25, which can be found in the full version, we first prune entries of the degree sequence with value 0 and then apply Lemma 22 together with Lemma 23. This pruning step is implemented implicitly by guessing the number $n_1 \in [1, s_0]$ of vertices with degree at least 1.

We now combine Lemma 25 and the Havel-Hakimi algorithm [17, 19] into a reconstruction algorithm for StarSubRec where we assume the constraint graphs $F_1, \dots, F_k$ to be exactly all stars of size up to k.

▶ **Theorem 26.** *The problem* StarSubRec *can be solved in time* $n^{O(k^2)}$ *if* $F_i = S_{i-1}$ *for all* $i \in [1, k]$ *where* n *is the count from the constraint* (S_0, n)*.*

Algorithm 1 $\mathrm{DP}(s_0, \ldots, s_\ell, m, x)$.

if $s_1 = 0$ **then** `// Base cases (part 1)`
  return $(s_2 = 0 \wedge \cdots \wedge s_\ell = 0)$ `// Empty graph or isolated nodes`
end
if $s_0 = 1$ **then** `// Base cases (part 2)`
  return $(s_1 = 0 \wedge \cdots \wedge s_\ell = 0) \vee (s_1 = 1 \wedge s_2 = 0 \wedge \cdots \wedge s_\ell = 0 \wedge m \geq 1 \wedge x \geq 1)$
end
for $d \leftarrow 1$ **to** m **do** `// Guess highest degree`
  for $n_1 \leftarrow 1$ **to** s_0 **do** `// Guess number of vertices with non-zero degree`
    if $d \leq (n_1 - 1) + x$ **then** `// Check realizability of highest degree`
      $s'_0 \leftarrow n_1 - 1$
      $s'_1 \leftarrow s_1 - \binom{d}{1} - s'_0$
      $\ldots$
      $s'_\ell \leftarrow s_\ell - \binom{d}{\ell} - s'_{\ell-1}$
      $m' \leftarrow d - 1$
      $x' \leftarrow x + n_1 - 1 - d$
      if $\mathrm{DP}(s'_0, \ldots, s'_\ell, m', x')$ **then** `// Recursion`
        return ***True***
      end
    end
  end
end
return ***False***

Proof. We give an algorithm $\mathcal{A}$ for StarSubRec. Let $(F_1, m_1), \ldots, (F_k, m_k)$ be the input of StarSubRec. We denote this input as $(S_0, s_0), \ldots, (S_\ell, s_\ell)$. Let $n := s_0$. First, $\mathcal{A}$ computes $\mathrm{DP}(s_0, s_1/2, s_2, \ldots, s_\ell, n-1, 0)$ using dynamic programming with the recursion from Lemma 25 and $\ell + 3$ table dimensions $s_0, \ldots, s_\ell, m, x$. The dynamic program always runs into a base case ($s_0 < 2$ or $s_1 < 1$) which is trivial to solve (see Algorithm 1) and therefore terminates because the parameters $s_0, \ldots, s_\ell, m$ all get smaller at least by 1 in each recursive call. This happens in runtime $n^{O(\ell^2)}$ because all parameters $s_0, \ldots, s_\ell, m, x$ are bounded by n^ℓ so there are at most $n^{O(\ell^2)}$ entries in the dynamic programming table that $\mathcal{A}$ might compute and in each recursion step $\mathcal{A}$ tries at most n^2 combinations of the values d and n_1. Algorithm 1 shows pseudo-code for the computation of $\mathrm{DP}(s_0, \ldots, s_\ell, m, x)$.

If $\mathrm{DP}(s_0, \ldots, s_\ell, n-1, 0) = 0$ then $\mathcal{A}$ outputs False. Otherwise $\mathcal{A}$ reconstructs the degree sequence $(d_1, \ldots, d_n)$ from the dynamic programming using backtracking. Specifically, the parameter d in the first recursive call from Lemma 25 corresponds to d_1. Similarly, the parameter d from the recursive call number i is d_i. Finally, $\mathcal{A}$ runs the Havel-Hakimi algorithm [17, 19] (also see [35]) to reconstruct a graph G with the degree sequence $(d_1, \ldots, d_n)$ in $O(n^2)$ time and then outputs G. ◀

As already mentioned we get the same result for homomorphisms if we use the recursion $h_j(d)$ from Lemma 20.

To prove Theorem 3 we have to deal with unspecified counts for stars S_j with $j < \ell$. Since they are all bounded by the largest homomorphism count, we can simply guess the "missing" counts and call $\mathcal{A}$ for each guess.

Proof of Theorem 3. We give an algorithm $\mathcal{B}$ for StarHomRec. Let $(F_1, m_1), \ldots, (F_k, m_k)$ be an instance of StarHomRec. Let $\ell := \max_{i \in [k]} |F_i| - 1$ and $m := \{m_i \mid i \in [k]\}$. Each F_i is a star S_j with $j \leq \ell$ but there might be stars S_j with $j < \ell$ for which there is no count specified. $\mathcal{B}$ guesses all "missing" star counts up to the star S_ℓ so that we get constraints $(S_0, s_0), \ldots, (S_\ell, s_\ell)$ where $s_j = m_i$ for all $i \in [0, k], j \in [0, \ell]$ with $S_j = F_i$. For each guess, $\mathcal{B}$ calls the homomorphism variant of algorithm $\mathcal{A}$ from the proof of Theorem 26 with input $(S_0, s_0), \ldots, (S_\ell, s_\ell)$.

$\mathcal{B}$ calls $\mathcal{A}$ at most m^ℓ times because it has to guess at most ℓ counts which are all bounded by $s_1 \leq s_2 \leq \cdots \leq s_\ell \leq m$ and we can also bound $s_0 \leq s_1$ because $s_0 > s_1$ would just correspond to isolated vertices which are trivial to add. Overall, $\mathcal{B}$ has a runtime of $m^\ell m^{O(k^2)} = m^{O(k^2)}$. ◀

6 Conclusions

We determine the exact computational complexity of the homomorphism reconstruction problem both in the case that the constraints are specified in binary and unary: the (binary) HomRec problem is NEXPTIME-complete, and the (unary) UnHomRec is Σ_2^p-complete.

Consider the following parameterized version of the unary reconstruction problem. It was proved in [4] that the binary version of this parameterized problem is hard.

> p-UnHomRec
> **Input:** Pairs $(F_1, m_1), \ldots, (F_k, m_k)$, where $F_1, \ldots, F_k$ are graphs and $m_1, \ldots, m_\ell \in \mathbb{N}$ (in unary encoding).
> **Parameter:** $\sum_{i=1}^k |F_i|$
> **Question:** Is there a graph G such that $\hom(F_i, G) = m_i$ for every $i \in [k]$?

It is open if this problem is fixed-parameter tractable, or at least the complexity class XP, that is, solvable in polynomial time for each fixed parameter value. We prove that the problem is in XP if the F_i are stars.

References

1 Miklos Ajtai, Ronald Fagin, and Larry J. Stockmeyer. The closure of monadic NP. *J. Comput. Syst. Sci.*, 60(3):660–716, 2000. doi:10.1006/jcss.1999.1691.

2 László Babai. Automorphism groups, isomorphism, reconstruction. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume 2, pages 1447–1540. MIT Press, 1995.

3 Jan Böker, Yijia Chen, Martin Grohe, and Gaurav Rattan. The Complexity of Homomorphism Indistinguishability. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2019.54.

4 Jan Böker, Louis Härtel, Nina Runde, Tim Seppelt, and Christoph Standke. The Complexity of Homomorphism Reconstructibility. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *Proceedings of the 41st International Symposium on Theoretical Aspects of Computer Science*, volume 289 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.19.

5 Jan Böker, Louis Härtel, Nina Runde, Tim Seppelt, and Christoph Standke. The complexity of homomorphism reconstructibility, 2023. doi:10.48550/arXiv.2310.09009.

6 Marek Černý and Tim Seppelt. Homomorphism indistinguishability, multiplicity automata equivalence, and polynomial identity testing. In *Proceedings of the 43rd International Symposium on Theoretical Aspects of Computer Science (this conference)*, LIPIcs, pages 25:1–25:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.STACS.2026.25.

7 A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

8 Surajit Chaudhuri and Moshe Y. Vardi. Optimization of Real Conjunctive Queries. In Catriel Beeri, editor, *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC, USA*, pages 59–70. ACM Press, 1993. doi:10.1145/153850.153856.

9 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms Are a Good Basis for Counting Small Subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 210–223. Association for Computing Machinery, 2017. event-place: Montreal, Canada. doi:10.1145/3055399.3055502.

10 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 40:1–40:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.40.

11 Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, August 2010. doi:10.1002/jgt.20461.

12 Paul Erdős and Tibor Gallai. Gráfok előírt fokszámú pontokkal (graphs with prescribed degrees of vertices). *Matematikai Lapok*, 11:264–274, 1960.

13 Hana Galperin and Avi Wigderson. Succinct representations of graphs. *Inf. Control.*, 56(3):183–198, 1983. doi:10.1016/S0019-9958(83)80004-7.

14 Martin Grohe. Counting Bounded Tree Depth Homomorphisms. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 507–520, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373718.3394739.

15 Martin Grohe. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 1–16. ACM, 2020. doi:10.1145/3375395.3387641.

16 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism tensors and linear equations. *Advances in Combinatorics*, 2025(4), 2025. doi:10.19086/aic.2025.4.

17 S. L. Hakimi. On the realizability of a set of integers as degrees of the vertices of a linear graph. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962. doi:10.1137/0110037.

18 Hamed Hatami and Serguei Norine. Undecidability of linear inequalities in graph homomorphism densities. *Journal of the American Mathematical Society*, 24(2):547–547, May 2011. doi:10.1090/S0894-0347-2010-00687-X.

19 Václav Havel. Poznámka o existenci konečných grafů (a remark on the existence of finite graphs). *Časopis pro pěstování matematiky*, 80(4):477–480, 1955. URL: https://eudml.org/doc/19050.

20 Emily Jin, Michael M. Bronstein, İsmail İlkan Ceylan, and Matthias Lanzinger. Homomorphism counts for graph neural networks: All about that basis. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=zRrzSLwNHQ.

21 Prem Nigam Kar, David E. Roberson, Tim Seppelt, and Peter Zeman. NPA hierarchy for quantum isomorphism and homomorphism indistinguishability. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, volume 334 of *LIPIcs*, pages 105:1–105:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.105.

22 Paul Joseph Kelly. A congruence theorem for trees. *Pacific Journal of Mathematics*, 7:961–968, 1957.

23 Ker-I Ko and Wen-Guey Tzeng. Three Σ_2^P-complete problems in computational learning theory. *computational complexity*, 1(3):269–310, 1991. `doi:10.1007/BF01200064`.

24 Swastik Kopparty and Benjamin Rossman. The homomorphism domination exponent. *European Journal of Combinatorics*, 32(7):1097–1114, 2011. Homomorphisms and Limits. `doi:10.1016/j.ejc.2011.03.009`.

25 László Lovász. *Large networks and graph limits.* Number volume 60 in American Mathematical Society colloquium publications. American Mathematical Society, Providence, Rhode Island, 2012. `doi:10.1090/coll/060`.

26 Lázló Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3):321–328, September 1967. `doi:10.1007/BF02280291`.

27 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672, 2020. `doi:10.1109/FOCS46700.2020.00067`.

28 Jerzy Marcinkowski and Mateusz Orda. Bag semantics conjunctive query containment. four small steps towards undecidability. *Proceedings of the ACM on Management of Data*, 2(2):1–24, 2024.

29 Jerzy Marcinkowski and Piotr Ostropolski-Nalewaja. Bag semantics query containment: The cq vs. ucq case and other stories. *ArXiv*, 2503.07219, 2025. `doi:10.48550/arXiv.2503.07219`.

30 P. V. O'Neil. Ulam's conjecture and graph reconstructions. *The American Mathematical Monthly*, 77(1):35–43, 1970. `doi:10.2307/2316851`.

31 Christos H. Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71(3):181–185, 1986. `doi:10.1016/S0019-9958(86)80009-2`.

32 Alexander A. Razborov. On the Minimal Density of Triangles in Graphs. *Combinatorics, Probability and Computing*, 17(4):603–618, July 2008. `doi:10.1017/S0963548308009085`.

33 Tim Seppelt. An algorithmic meta theorem for homomorphism indistinguishability. In Rastislav Královic and Antonín Kucera, editors, *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024, Bratislava, Slovakia, August 26-30, 2024*, volume 306 of *LIPIcs*, pages 82:1–82:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPIcs.MFCS.2024.82`.

34 Tim Seppelt. Logical equivalences, homomorphism indistinguishability, and forbidden minors. *Information and Computation*, 301:105224, 2024. `doi:10.1016/j.ic.2024.105224`.

35 Shahriar Shahriari. *An invitation to combinatorics.* Cambridge University Press, 2021.

36 Gottfried Tinhofer. A note on compact graphs. *Discrete Applied Mathematics*, 30(2-3):253–264, 1991. `doi:10.1016/0166-218X(91)90049-3`.

37 Stanisław Ulam. *A collection of mathematical problems.* Wiley, 1960.

38 Hinrikus Wolf, Luca Oeljeklaus, Pascal Kühner, and Martin Grohe. Structural node embeddings with homomorphism counts. *ArXiv*, arXiv:2308.15283, 2023. `doi:10.48550/arXiv.2308.15283`.