Caterpillar GNN: Replacing Message Passing with Graph-Level Aggregation

Marek Černý
University of Antwerp, Belgium
marek.cerny@uantwerp.be

Abstract

Message-passing graph neural networks (MPGNNs) dominate modern graph learning. Typical efforts enhance MPGNN's expressive power by enriching the adjacency-based aggregation. In contrast, we introduce a *graph-level aggregation* over walk incidence-based matrices that are constructed to deliberately trade off some expressivity for stronger and more structured inductive bias. This approach allows for gradual scaling between classical message-passing and simpler methods based on walks. Drawing on *weighted automata*, we ensure our gradual scaling is polynomial-time and avoids the combinatorial explosion of colored walks. Our second result characterizes the expressive power at each scale using homomorphism counts over a hierarchy of *generalized caterpillar graphs*. Based on these foundations, we propose Caterpillar GNNs that substantially reduce the number of nodes in the hidden layers of the computational graphs on real-world datasets. This gradual reduction does not obstruct learning, enabling a systematic study of lower-order expressivity. We further describe benchmark settings where task-aligned expressivity supports learning.

1 Introduction

Graphs are a powerful structure, capable of representing relational information across various domains such as biology, chemistry, databases, or social sciences. Unlike sequences or grids, graphs do not come with a fixed topology of computation: the topology is part of the input. The established incorporation of this variability relies on the inductive bias of (equivariant) message-passing (MP) in graph neural networks (MPGNNs). Prior work has shown the limits of MP in capturing structural biases [Xu et al., 2019, Morris et al., 2019], leading to many extensions of MP. Learning is also affected by phenomena such as nodal over-smoothing [Oono and Suzuki, 2020] and over-squashing [Alon and Yahav, 2021], motivating direct interventions in neighborhood topology [Topping et al., 2022, Di Giovanni et al., 2023].

For graph-level learning, the usefulness of a structural bias depends on the task. Our motivation is expressivity alignment: retaining structure suited to the patterns relevant for prediction, while keeping the training procedure undisturbed by unaligned biases. In this work, we develop foundations for tools that can diagnose potentially harmful effects of task-unaligned expressivity. Our approach is to control the structure available to the neural architecture and study the resulting inductive biases. We achieve this through an exact characterization of lower-order expressivity and a polynomial-time construction of a computational graph that is incorporated into a learnable architecture. This shows a complementary way to organize the computation around expressivity alignment by controlling the structural information embodied in the aggregation matrices. The proposed set of results provides a controlled inductive bias that can be empirically studied in standard GNN architectures.

Our starting point is colored-walk statistics. A colored walk is a sequence, or a word, where each letter represents a vertex color visited along a walk. Its weight is the number of its occurrences in

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

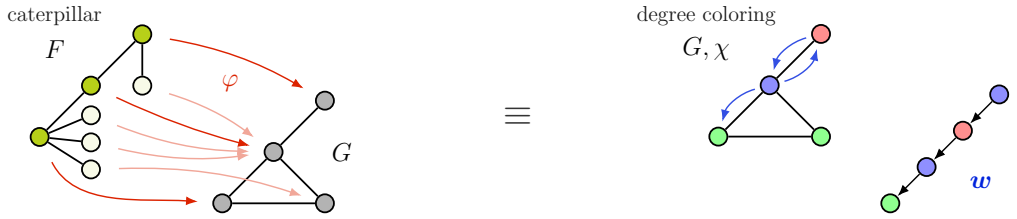

Figure 1: Counting over either of the two depicted notions is exactly as expressive on graphs. Left: *A graph homomorphism* $\varphi\colon F \to G$, where F is a caterpillar graph. Right: *An occurrence of a colored walk* $\boldsymbol{w} =$ brbg in the same graph G colored by $\chi_{\deg}$ according to vertex degrees.

the input graph. With a constant vertex coloring, this weight reduces to the ordinary walk count. Coloring vertices by their degree additionally records degree patterns. We iteratively refine the coloring using the color refinement [Babai and Kučera, 1979]. Each refinement round can attach richer local structure to each letter, up to the full limit of MP's expressivity.

The number of distinct colored walks grows exponentially with their length in the worst case. As a result, existing architectures process sequential patterns using random-walk sampling [Tönshoff et al., 2023, Zeng et al., 2023, Chen et al., 2024]. Since random sampling sacrifices equivariance, we instead construct a canonical representation of colored-walk counts. For each walk length, a walk-incidence matrix records occurrences of colored walks terminating at each vertex. Since each column has one entry per vertex, its column space has dimension at most the number of input vertices. The subsequent canonical reduction of this column space draws on the techniques of *weighted (or multiplicity) automata* [Schützenberger, 1961, Tzeng, 1996], recognizing a weighted counterpart of regular languages. The layer-specific reduced matrices define our *graph-level aggregation* (GLA), constructed in polynomial time while retaining the expressive power of the colored-walk statistics. We instantiate GLA in Caterpillar GNNs that enable a systematic study of learning on graphs under controlled lower expressivity over a fixed set of hyperparameters.

To understand and characterize the graph structure accessible through GLA, or equivalently through colored walks, we use an algebraic definition of expressive power. A graph homomorphism maps the vertices of a source graph to the input graph while preserving adjacency. Choosing a class of source graphs, denoted by $\mathcal{F}$, gives an invariant by counting these maps from every graph in $\mathcal{F}$. The expressivity of some MPGNN extensions can be characterized by graph classes $\mathcal{F}$ that contain all trees, see Table 1. In the limit, extending $\mathcal{F}$ from trees to all finite graphs yields graph isomorphism, as shown by Lovász [1967]. *Our work answers the complementary question: which inductive biases arise when* $\mathcal{F}$ *is restricted downwards—to strict subclasses of trees?*

To that end, we propose a suitable hierarchy of *generalized caterpillar graphs*—trees that yield a path after a bounded number of leaf-removal rounds. We call this bound *height*, in particular, height one corresponds to one round of removing all leaves, and defines folklore caterpillars. We completely characterize expressivity of this height-controlled hierarchy by counting walks colored by a correspondingly bounded number of color refinement rounds. In particular, for an input graph G, counting homomorphisms to G over folklore caterpillars is exactly as expressive as coloring G according to vertex degrees and then counting colored walks. See Figure 1.

Higher order[b]	treewidth[g]
$\mathcal{P}$-enabled[c]	$\mathcal{P}$-trees[c]
Subgraph Aggr.[d,e]	apex trees[h]
Spectral Invariant[f]	parallel trees[i]
Vanilla MPGNN[a]	trees[g]
Caterpillar (ours)	caterpillars

Table 1: MPGNNs and corresp. expressivity bounds using class $\mathcal{F}$. [a]Gilmer et al. [2017], [b]Morris et al. [2019], [c]Barceló et al. [2021], [d]Qian et al. [2022] [e]Frasca et al. [2022], [f]Zhang et al. [2024], [g]Dvořák [2010], [h]Rattan and Seppelt [2021], [i]Gai et al. [2025].

GLA thus provides a homomorphism canonization over generalized caterpillars of bounded height. Within the graph class determined by this correspondence, its invariant is sufficient to identify graphs up to isomorphism, as follows from the factorization argument of Lovász [1967]. This motivates applications where the task depends on caterpillar-like structures such as proteins or polymers. Finally, we construct a dataset, NStepAddition, benchmarking this identification ability by construction: we encode addition of two binary numbers as a small unlabeled caterpillar-like graph. Solving it only requires expressive power of homomorphism counts over bounded-height caterpillars, and helps to empirically uncover settings where MP tends to be task-unaligned and suffering from topological bottlenecks.

Our main contributions are as follows.

1. We introduce GLA (Section 3). We prove its tractability (Theorem 3.1) and desired expressivity (Theorem 4.2). The challenge of its complete derivation and proofs we address by developing techniques in automata theory (Appendix A).
2. We characterize expressivity of GLA using a hierarchy of *generalized caterpillar graphs* and its graph homomorphism counts (Section 4, Theorem 4.1). For this, we develop novel combinatorial arguments in graph theory (Appendix B).
3. We incorporate GLA into Caterpillar GNNs (Section 3.2, Eq. (4)), and investigate its gradual scaling (Section 3.3). Using walk incidence-based topology (Section 5.1), we illustrate how the optimal expressivity can be important when its inductive bias aligns with the task.

Empirically, on TUDataset, we illustrate dataset-specific tradeoffs between accuracy and nodal efficiency of gradual lower-order expressivity of Caterpillar GNN. Under practical conditions, GLA substantially reduces the number of nodes in the hidden layers of the computational graphs on datasets from Open Graph Benchmark [Hu et al., 2021], LRGB [Dwivedi et al., 2022], ZINC [Irwin et al., 2012], MoleculeNet [Wu et al., 2018], and MalNet-Tiny [Freitas et al., 2021].

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with a finite vertex set V and an edge set $E \subseteq V^2$. Loops are not assumed, and an edge between u and v is denoted by uv. The degree of a vertex u is $\deg(u) = |\{v \mid uv \text{ in } E\}|$, and $n = |V|$. A path is a connected acyclic graph with vertices of degree at most two. We denote the class of all paths by $\mathcal{P}$, and by $\mathcal{P}_t \subset \mathcal{P}$ the subclass of paths of length at most t where length means $|E|$. A tree is a connected acyclic graph. We denote by $\mathcal{T}$ the class of all trees. By $\mathcal{T}^\bullet$, we mean the class of rooted trees, and by $\mathcal{T}_h^\bullet \subset \mathcal{T}^\bullet$, the class where every vertex is at distance at most h from the root, that is, at most h edges from a root (e.g. [Diestel, 2025, page 8]).

Multisets are represented using symbols $\{\!\{$, and $\}\!\}$. Let X, Y be sets, and x in X, y in Y. The family of all multisets of elements from X is denoted by $\mathbb{N}^X$. For a multiset m in $\mathbb{N}^X$, we access multiplicity of x by $m[x]$. For a vector $\boldsymbol{v}$ in $\mathbb{R}^X$, we access its x-th component by $\boldsymbol{v}[x]$. For a matrix $\boldsymbol{M}$ in $\mathbb{R}^{X\times Y}$ (of shape $X \times Y$), we access its entries by $\boldsymbol{M}[x, y]$, rows by $\boldsymbol{M}[x]$ and columns by $\boldsymbol{M}[-, y]$. Finally, the notation $[k]$ stands for the set $\{1, 2, \ldots, k\}$ for $k \in \mathbb{N}$. For the graph G, we denote its *adjacency matrix* by $\boldsymbol{A}$ in $\mathbb{R}^{V\times V}$, that is, $\boldsymbol{A}[u, v] = 1$ if uv in E and 0 otherwise. Its *identity* or *self-loop matrix* is denoted by $\boldsymbol{I}$ in $\mathbb{R}^{V\times V}$, and the all-ones vector by $\mathbf{1}$ in $\mathbb{R}^{V\times 1}$.

MPGNNs. Throughout, we often represent graphs by matrices, for which we adopt a specific notation. For a matrix $\boldsymbol{M}$ in $\mathbb{R}^{X\times Y}$, features of d channels $\boldsymbol{h}$ in $\mathbb{R}^{Y\times d}$ and x in X, we define

$$\mathrm{mult}(x, \boldsymbol{M}, \boldsymbol{h}) \coloneqq \{\!\{\boldsymbol{M}[x, y] \cdot \boldsymbol{h}[y] \mid y \text{ in } Y\}\!\}.$$

Let $\boldsymbol{h}_{\text{MP}}^{(0)}$ be features in $\mathbb{R}^{V\times d}$ if given and $\mathbf{1}$ otherwise. Then MPGNN of L layers is given for each u in V and integer ℓ such that $0 \leqslant \ell < L$ as follows

$$\boldsymbol{h}_{\text{MP}}^{(\ell+1)}[u] = \mathrm{UPDATE}\left(\mathrm{mult}(u, \boldsymbol{I}, \boldsymbol{h}_{\text{MP}}^{(\ell)}),\ \mathrm{AGGREGATE}\left(\mathrm{mult}(u, \boldsymbol{A}, \boldsymbol{h}_{\text{MP}}^{(\ell)})\right)\right),$$
$$\boldsymbol{h}_{\text{MP}} = \mathrm{READOUT}\left(\mathrm{mult}(0, \tfrac{1}{n}\mathbf{1}^\top, \boldsymbol{h}_{\text{MP}}^{(L)})\right),$$

where we use mult to self-loop with $\boldsymbol{I}$, to range over adjacent nodes with $\boldsymbol{A}$ of shape $V \times V$ and to collect all nodes with $\frac{1}{n}\mathbf{1}^\top$ of shape $\{0\} \times V$. The functions AGGREGATE, UPDATE, and READOUT are specific to each layer. We omit their indexing and learnable parameters.

Expressivity and homomorphism counts. Let $\mathcal{G}$ denote the class of all graphs and let f and g be two functions on $\mathcal{G}$. Then the function f *is at least as expressive as* g, denoted by $\mathsf{f} \sqsupseteq \mathsf{g}$, if for every two graphs G and H, it holds that $\mathsf{f}(G) = \mathsf{f}(H)$ implies $\mathsf{g}(G) = \mathsf{g}(H)$. Next, f *is (exactly) as expressive as* g, denoted by $\mathsf{f} \equiv \mathsf{g}$, if $\mathsf{f} \sqsupseteq \mathsf{g}$ and $\mathsf{g} \sqsupseteq \mathsf{f}$. Furthermore, f *is (strictly) more expressive than* g, denoted by $\mathsf{f} \sqsupsetneq \mathsf{g}$, if $\mathsf{f} \sqsupseteq \mathsf{g}$ and $\mathsf{f} \not\equiv \mathsf{g}$. The relation of expressivity is a partial ordering on the family of functions on $\mathcal{G}$.

For a source graph $F = (V_s, E_s)$, a function $\varphi\colon V_s \to V$ is a *graph homomorphism* $F \to G$ if every edge uv in E_s implies edge $\varphi(u)\varphi(v)$ in E. See Figure 1. For a class of source graphs $\mathcal{F} \subseteq \mathcal{G}$, we

define a (possibly infinite) vector of *homomorphism counts over* $\mathcal{F}$, denoted by $\mathsf{hom}(\mathcal{F}, G)$ in $\mathbb{N}^{\mathcal{F}}$, as $\mathsf{hom}(\mathcal{F}, G)[F] = |\{\varphi \mid \varphi\colon F \to G\}|$ for all F in $\mathcal{F}$. Note that every class $\mathcal{F} \subseteq \mathcal{G}$ induces the function $\mathsf{hom}(\mathcal{F}, -)\colon \mathcal{G} \to \mathbb{N}^{\mathcal{F}}$, assigning $\mathsf{hom}(\mathcal{F}, G)$ to the target graph G. It always holds that $\mathcal{F} \supseteq \mathcal{F}'$ implies $\mathsf{hom}(\mathcal{F}, -) \sqsupseteq \mathsf{hom}(\mathcal{F}', -)$.

Graph colorings and color refinement. A coloring χ is a map that assigns specific colors to the vertices. Formally, for each graph $G = (V, E)$, we have a function $\chi(G, -)\colon V \to \Sigma'$ where Σ' denotes a color set. We say coloring χ is a Σ-*coloring on* G if $\Sigma' \subseteq \Sigma$. A coloring example is the trivial coloring χ_{triv}, which assigns 0 to every vertex u of G, $\chi_{\mathrm{triv}}(G, u) = 0$. Therefore, χ_{triv} is a $\{0\}$-coloring on every graph. Another "extreme" is the *identity coloring* χ_{id} assigning identities on vertices, $\chi_{\mathrm{id}}(G, u) = u$ for vertex u, and thus χ_{id} is a V-coloring on G. The *degree coloring* $\chi_{\deg}$, assigns degree to every vertex in G which can be written as $\chi_{\deg}(G, -) = \deg(-)$.

A *color refinement* constructs a sequence of graph colorings: $\chi_{\mathsf{cr}}^{(0)}(G, u) = 1$, and for all $h \geqslant 0$ and each u in V as $\chi_{\mathsf{cr}}^{(h+1)}(G, u) = (\chi_{\mathsf{cr}}^{(h)}(G, u), \{\!\{\chi_{\mathsf{cr}}^{(h)}(G, v) \mid uv \text{ in } E\}\!\})$. Secondly, it defines a sequence of functions on graphs $\mathsf{cr}^{(h)}(G) = \{\!\{\chi_{\mathsf{cr}}^{(h)}(G, u) \mid u \text{ in } V\}\!\}$; and, finally, the function on graphs: $\mathsf{cr}(G) = \{\mathsf{cr}^{(h)}(G) \mid h \text{ in } \mathbb{N}\}$. For instance, the first coloring is as expressive as the trivial: $\chi_{\mathsf{cr}}^{(0)}(G, -) \equiv \chi_{\mathrm{triv}}(G, -)$, and the second exactly as the degree coloring: $\chi_{\mathsf{cr}}^{(1)}(G, -) \equiv \chi_{\deg}(G, -)$.

Let χ be a Σ-coloring on G. A *walk* in G is a sequence of vertices $v_1, v_2, \ldots, v_t$ such that $v_i v_{i+1}$ in E for i in $[t-1]$. A special case is a *path* in G, which is a walk with all vertices distinct. A *colored walk* is a word $\boldsymbol{a} = a_1 a_2 \ldots a_t$ such that $a_i = \chi(G, v_i)$ in Σ for i in $[t]$. At the same time, the sequence $v_1, v_2, \ldots, v_t$ is an *occurrence* of $\boldsymbol{a}$ in G. See Figure 1. We say that vertex u is *incident* to colored walk $\boldsymbol{a}$ if $u = v_t$, and *adjacent* if $uv_t \in E$. We denote by Σ^t, resp. $\Sigma^{\leqslant t}$ the set of all words over Σ of length exactly t, resp. at most t; and by Σ^* the set of all words. Note that $\Sigma^0 = \{\lambda\}$ where λ is the *empty word*.

3 Constructing Graph-Level Aggregation

This section introduces graph-level aggregation (GLA) and its instantiation in Caterpillar GNNs. Although we start from colored walks, the main idea is to avoid one-by-one sequential processing. For each length t, we collect colored-walk statistics in a walk-incidence matrix $\boldsymbol{W}_t$. Since the number of possible colored walks grows exponentially, we select a canonical linearly independent subset of columns, yielding a reduced space of dimension at most linear in the number of vertices of the graph. We then project and restrict the adjacency and identity matrices onto consecutive layer-specific reduced spaces. The resulting matrices form the layer-specific computational graph of GLA.

Part I formalizes the canonical reduction of colored-walk statistics, Part II exploits this reduction to construct matrices defining GLA and its instantiation in Caterpillar GNNs. Part III explains how a single height parameter h can control the tradeoff between reduction and expressivity. Omitted proofs are given in Appendix A.

3.1 Part I: Sequential patterns

For tractable incorporation of lower-order inductive biases, we improve a way of processing sequential patterns further motivated by Theorem 4.1 as analyzed in Section 4. In the language of colored walks, many successful machine learning approaches first *sample* a tractable number of random walks and then process the visited colors as sequences with either kernels [Borgwardt et al., 2005, Kriege, 2022] or neural networks [Tönshoff et al., 2023, Zeng et al., 2023,

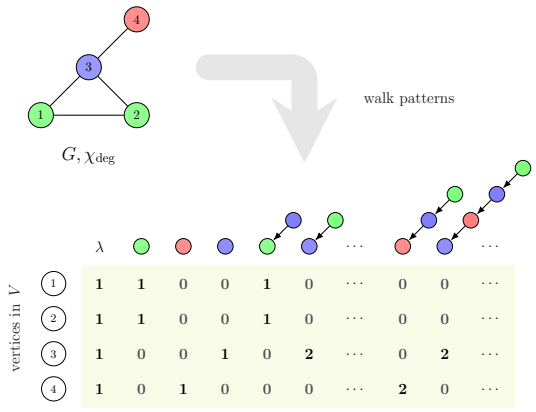

Figure 2: Top: Graph G with vertices colored by $\chi_{\deg}$. Colors red, green and blue depict degrees 1, 2 and 3, respectively. Bottom: Walk incidence matrix $\boldsymbol{W}$ of shape $V \times \Sigma^*$. The entry $\boldsymbol{W}[3, \mathrm{gb}]$ equals 2, since vertex 3 terminates two occurrences of the colored walk gb in G.

Chen et al., 2024]. Our approach is a fundamental *reversal* of these steps: given a prescribed colored walk, we count its occurrences.

Crucially, we consider a tractable and canonical subset of colored walks. As shown later (Theorem 4.2), this subset suffices to determine all other colored walks. In contrast to prior sampling-heavy methods, we preserve determinism, equivariance and intended expressivity. To formalize our reversal, we relate vertices and colored walks using incidence matrix.

Walk incidences. For a given graph G with χ a Σ-coloring, and a given length $t \geqslant 0$, we define the *walk-incidence matrix* $\boldsymbol{W}_t$ of shape $V \times \Sigma^t$ for each u in V and $\boldsymbol{a}$ in Σ^t by the entry

$$\boldsymbol{W}_t[u, \boldsymbol{a}] \tag{1}$$

that equals the number of occurrences of colored walk $\boldsymbol{a}$ terminating in vertex u. Each column $\boldsymbol{W}_t[-, \boldsymbol{a}]$ in $\mathbb{N}^V \subseteq \mathbb{R}^V$ corresponds to a *multiset of vertices* incident to colored walk $\boldsymbol{a}$. For instance, the column $\boldsymbol{W}_1[-, c]$ coincides with vertices u of color $c = \chi(G, u)$ in Σ. By *convention*, the empty walk is incident to every vertex, $\boldsymbol{W}_0[u, \lambda] = 1$ for u in V. See the (infinite) matrix $\boldsymbol{W} = [\boldsymbol{W}_0 | \boldsymbol{W}_1 | \cdots]$ of shape $V \times \Sigma^*$ in Figure 2.

Walk selection. The row dimension V of incidence matrices $\boldsymbol{W}_t$ remains fixed, while the column dimension Σ^t grows exponentially in t. We avoid this growth by selecting subsets of Σ^t, such that the induced columns of $\boldsymbol{W}_t$ form a basis of the column space of $\boldsymbol{W}_t$. The definition proceeds inductively: $S_0 = \{\lambda\} = \Sigma^0$, and for known S_t, the set $S_{t+1} \subseteq \Sigma^{t+1}$ satisfies the following conditions:

(i) for every $\boldsymbol{a}c$ in S_{t+1} there is $\boldsymbol{a}$ in S_t (*prefix-closedness*),

(ii) the columns of $\boldsymbol{W}_{t+1}$ induced by S_{t+1} are *linearly independent*, and

(iii) the set S_{t+1} is *lexicographically minimal* among other sets satisfying (i) and (ii).

The last Condition (iii) together with $S_0 = \{\lambda\}$ ensures uniqueness, making this selection canonical. Condition (ii) implies the upper bound $|S_t| \leqslant \operatorname{rank}(\boldsymbol{W}_t) \leqslant |V|$, reducing the representation of potentially $|V|^t$ columns to at most $|V|$. Finally, Condition (i) allows for a time-efficient algorithm reminiscent of breadth-first search with linear independence checking using matrix decomposition.

Theorem 3.1. *Let χ be a Σ-coloring on a graph G with n vertices, and let T in $\mathbb{N}$ be a limit. Then the canonical subsets $(S_t)_{t=0}^{T}$ defined above are computable in time $\mathcal{O}(Tn^{\omega+\epsilon})$ for any $\epsilon > 0$ where ω is the exponent of matrix multiplication.*[1]

Exponential colored-walk statistics can be compressed to at most linear statistics. Three selection rules ensure the compression agrees on graphs with the same statistics, and it can be computed with the asymptotics of graph spectra estimation via T QR decompositions.

3.2 Part II: Reduced Matrices

Up to this point, we have considered walks as sequences processed one by one. However, such a representation is inefficient, in particular because it ignores shared prefix structure, as is well-known from string-searching algorithms (e.g., suffix trees [Weiner, 1973]). To overcome this inefficiency, we organize walk-incidence statistics into matrices, where selected colored walks correspond to columns and also rows. This matrix formulation enables hierarchical aggregation of walk patterns, analogous to how message passing (MP) aggregates over neighborhoods, but dropping the assumption of repeating fixed neighborhood structure at each layer.

MP on graph $G = (V, E)$ with n vertices consists of two steps: aggregation via adjacency operator $\boldsymbol{A}$ and update via self-looping operator $\boldsymbol{I}$ both $\mathbb{R}^V \to \mathbb{R}^V$. We aim to deliberately restrict this repeating mechanism: informally, we lock the corresponding vector space $\mathbb{R}^V$ by projecting those operators into $\mathbb{R}^{S_{t+1}} \to \mathbb{R}^{S_t}$ of possibly lower dimension as implied by Condition (ii).

For a Σ-coloring on G, and integer $t \geqslant 0$, we denote by $\boldsymbol{B}_t$ of shape $V \times S_t$ the submatrix of $\boldsymbol{W}_t$ (of shape $V \times \Sigma^t$) that keeps only the columns indexed by S_t (see Figure 3 for an illustration). Condition (ii) guarantees that every matrix $\boldsymbol{B}_t$ is tractable and has full rank.

[1] As of 2026, $\omega \leqslant 2.371339$ [Alman et al., 2024]. For instance, we can assume that $\omega + \epsilon \leqslant 2.372$.

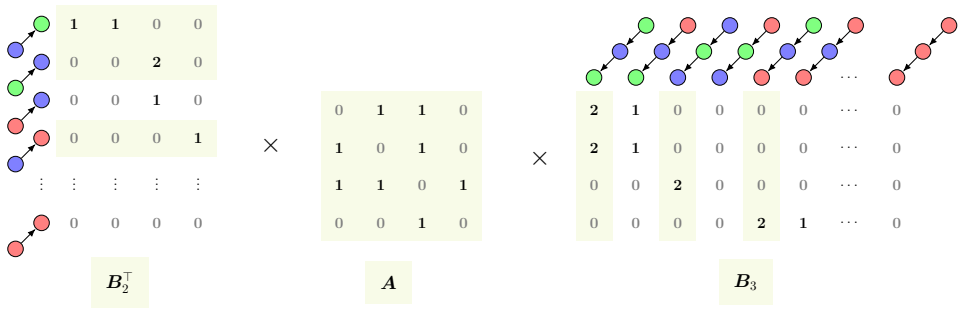

Figure 3: Locking the adjacency matrix $\boldsymbol{A}$ of the graph G as in Figure 2. Matrix $\boldsymbol{B}_3$ of shape $V \times S_3$ is a submatrix of $\boldsymbol{W}_3$ (of shape $V \times \Sigma^3$) induced by highlighted columns, similarly, $\boldsymbol{B}_2^\top$ and $\boldsymbol{W}_2^\top$.

Graph-Level aggregation. Let $\boldsymbol{M}$ be a matrix of shape $V \times V$, such as $\boldsymbol{A}$ or $\boldsymbol{I}$ above, then a *t-th reduced $\boldsymbol{M}$-matrix* $\boldsymbol{C}_t^{\boldsymbol{M}}$ of shape $S_t \times S_{t+1}$ is defined as

$$\boldsymbol{C}_t^{\boldsymbol{M}} = (\boldsymbol{B}_t^\top \boldsymbol{B}_t)^{-1} \boldsymbol{B}_t^\top \, \boldsymbol{M} \, \boldsymbol{B}_{t+1}, \tag{2}$$

which solves the least-squares problem $\arg\min_{\boldsymbol{C}} \| \boldsymbol{B}_t \boldsymbol{C} - \boldsymbol{M} \boldsymbol{B}_{t+1} \|_F$. Informally, the unique operator $\boldsymbol{C}_t^{\boldsymbol{M}} : \mathbb{R}^{S_{t+1}} \to \mathbb{R}^{S_t}$ is the best approximation of $\boldsymbol{M}$ in the basis indexed by canonical S_t.

We call a *graph-level aggregation (GLA)* the collection of the first n reduced adjacency and identity matrices into the graph invariant

$$\mathcal{I}_{\mathrm{R}}(G, \chi) = \left\{ \left(\boldsymbol{C}_t^{\boldsymbol{A}}, \boldsymbol{C}_t^{\boldsymbol{I}} \right) \mid 0 \leqslant t < n \right\}. \tag{3}$$

Since reduced matrices are indexed by colored walks, we compare $\mathcal{I}_{\mathrm{R}}$ directly across graphs. Its expressive power (Section 4) as the function on graphs $\mathcal{I}_{\mathrm{R}}(-, \chi)$ motivates the following model.

Caterpillar GNNs. We now describe how reduced matrices are used across L layers. *Caterpillar GNN* initializes with $\boldsymbol{h}_{\mathrm{GLA}}^{(0)}(\boldsymbol{a}c) = \mathrm{REDUCE}(c, \{\!\{\}\!\}, \mathbf{1})$ for colored walk $\boldsymbol{a}c$ in S_L. Then, at each layer ℓ such that $0 \leqslant \ell \leqslant L-1$ with $t_\ell = L - \ell$ and for each colored walk $\boldsymbol{a}c$ in S_{t_ℓ} we have

$$\begin{aligned} \boldsymbol{h}_{\mathrm{GLA}}^{(\ell+1)}[\boldsymbol{a}c] &= \mathrm{REDUCE}\left(c, \, \mathrm{mult}(\boldsymbol{a}c, \boldsymbol{C}_{t_\ell}^{\boldsymbol{I}}, \boldsymbol{h}_{\mathrm{GLA}}^{(\ell)}), \, \mathrm{AGGREGATE}\left(\mathrm{mult}(\boldsymbol{a}c, \boldsymbol{C}_{t_\ell}^{\boldsymbol{A}}, \boldsymbol{h}_{\mathrm{GLA}}^{(\ell)}) \right) \right), \\ \boldsymbol{h}_{\mathrm{GLA}} &= \mathrm{READOUT}\left(\mathrm{mult}(\lambda, \boldsymbol{C}_0^{\boldsymbol{I}}, \boldsymbol{h}_{\mathrm{GLA}}^{(L)}) \right), \end{aligned} \tag{4}$$

where the function REDUCE replaces the usual UPDATE and targets a colored walk $\boldsymbol{a}c$ instead of a fixed vertex and requires color c as an additional input. A visual side-by-side comparison with the standard MP is given in Figure 4.

> Reduced walk statistics provide the subspaces onto which message-passing matrices are projected and restricted. Using these reduced matrices, GLA defines its layer-specific computational graph. The simpler the initial statistics, the smaller these matrices become.

3.3 Part III: Gradual Lower-Order Scaling

The vertex coloring χ controls the coarseness of distinguished colored walks and thus governs the resulting inductive bias of GLA. In our approach to GLA, we utilize colorings of the color refinement $\chi = \chi_{\mathsf{cr}}^{(h)}$, simplifying the choice for end-users to the parameter $h \geqslant 0$ called *height*. To guide our exploration, we analyze two extreme cases: trivial coloring χ_{triv} and identity coloring χ_{id}. For illustration of a resulting reduced computational graph, consult Figure 4 for the case of graph in Figure 2, and Figure 13 for the case of a protein graph.

Under χ_{triv}, all vertices share the same color 0. Thus, every walk of length t has color $\boldsymbol{z}_t = 00\cdots0$ (constant word of length t). Hence, every set S_t collapses to the singleton of $\boldsymbol{z}_t$, and computation over L layers collapses to a linear sequence of length L. The REDUCE function receives the color 0 together with multisets of form $\mathrm{mult}(\boldsymbol{z}_t, \boldsymbol{C}, \boldsymbol{h})$ containing a single pair $(m, \boldsymbol{h}[\boldsymbol{z}_t])$, where m is a normalized count of plain walks (cf., walk partition [Chung, 1997]). Formally, we have:

Observation 3.2. *Let χ_{triv} be the $\{0\}$-coloring on a graph G with at least one edge. Then for every $t \geqslant 0$: (a) it holds that $|S_t| = 1$; (b) the only entries, $\boldsymbol{C}_t^{\boldsymbol{I}}[\boldsymbol{z}_t, \boldsymbol{z}_{t+1}]$ and $\boldsymbol{C}_t^{\boldsymbol{A}}[\boldsymbol{z}_t, \boldsymbol{z}_{t+1}]$ are equal to $\frac{\mathbf{1}^\top \boldsymbol{A}^{2t+1}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1}}$, and $\frac{\mathbf{1}^\top \boldsymbol{A}^{2t+2}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1}}$, respectively.*

When each vertex is assigned a distinct color under χ_{id}, every colored walk in the graph has its unique occurrence. Hence, every set S_t reaches the maximum size $|V|$, with one colored walk per vertex. In this regime, the reduced matrices coincide entry-wise with the original matrices. Moreover, if REDUCE ignores its first parameter then GLA reaches the computational graph of standard message passing.

Proposition 3.3. *Let χ_{id} be the V-coloring on a graph $G = (V, E)$ with n vertices. By $\boldsymbol{v}_{t,u}$, we denote the (unique) word in S_t with the last color u in V. Then for every $t \geqslant 1$: (a) it holds that $|S_0| = 1$, and $|S_t| = |V|$; (b) for entries $\boldsymbol{C}_0^{\boldsymbol{I}}[\lambda, u] = \frac{1}{n}$, and $\boldsymbol{C}_t^{\boldsymbol{I}}[\boldsymbol{v}_{t,u}, \boldsymbol{v}_{t+1,v}] = \boldsymbol{I}[u, v]$ and $\boldsymbol{C}_t^{\boldsymbol{A}}[\boldsymbol{v}_{t,u}, \boldsymbol{v}_{t+1,v}] = \boldsymbol{A}[u, v]$.*

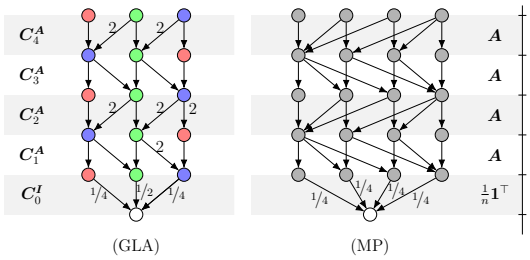

Figure 4: Comparison of computational graphs (without self-loops): (GLA) *graph-level aggregation* (ours), and (MP) *message-passing* for the graph G and coloring χ_{deg} as given in Figure 2 (left). Connections between layers are given by (GLA) t-th reduced graph matrices; (MP) copies of the adjacency matrix and the global readout. For unit weights, we omit labels.

The *height parameter* h controls the tradeoff between per-layer reduction and expressivity. Smaller h yields smaller reduced layers, down to a single-node layer weighted by plain walk counts at $h = 0$. Larger h preserves more of the structural information, up to message passing.

$$\begin{array}{ccccccccccc}
\mathsf{hom}(\mathcal{P}, -) & \subsetneq & \mathsf{hom}(\mathcal{C}_1, -) & \subsetneq & \mathsf{hom}(\mathcal{C}_2, -) & \subsetneq \cdots \subsetneq & \mathsf{hom}(\mathcal{C}_h, -) & \subsetneq \cdots \subsetneq & \mathsf{hom}(\mathcal{T}, -) \\
\| & & \| & & \| & & \| & & \| \\
\mathcal{I}_{\mathsf{R}}(-, \chi_{\text{triv}}) & \subsetneq & \mathcal{I}_{\mathsf{R}}(-, \chi_{\text{deg}}) & \subsetneq & \mathcal{I}_{\mathsf{R}}(-, \chi_{\mathsf{cr}}^{(2)}) & \subsetneq \cdots \subsetneq & \mathcal{I}_{\mathsf{R}}(-, \chi_{\mathsf{cr}}^{(h)}) & \subsetneq \cdots \subsetneq & \mathsf{cr}(-)
\end{array}$$

Figure 5: *Strictly separated lower-order expressivity hierarchy* of graph-level aggregations $\mathcal{I}_{\mathsf{R}}$ along the height parameter h related to caterpillar graph classes $\mathcal{P} = \mathcal{C}_0 \subset \mathcal{C}_1 \subset \cdots \subset \mathcal{C}_h \subset \cdots \subset \mathcal{T}$.

4 Characterizing Expressivity of GLA using Caterpillar Hierarchy

This section characterizes the expressivity of GLA (Equation 3) using graph homomorphism counts, providing a *scale* of expressive power of the associated lower-order inductive biases, see Figure 5. Our two main results, Theorem 4.1 and Theorem 4.2, established separately in the respective subsections, together give the vertical equivalences. The last vertical equivalence involving $\mathcal{T}$ is due to Dvořák [2010, Theorem 7]. Note that color refinement cr symbolizes message-passing (MP). The strictness of horizontal bounds follows from Lemma B.13, adopting the recent results of Roberson [2022], Schindling [2025], Neuen and Seppelt [2026]. We proceed with formal definitions.

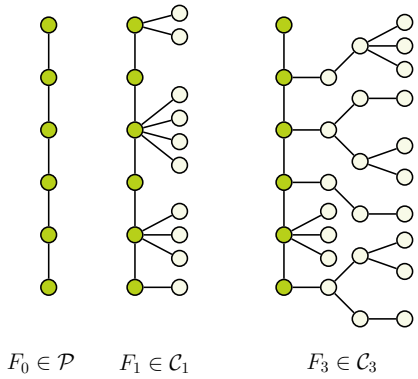

Figure 6: Caterpillar graphs with highlighted (possible) spine (green). Graph F_0 is a path of length 6 and hence a $(0, 6)$-caterpillar. Graph F_1 is a $(1, 6)$-caterpillar, and F_3 is a $(3, 6)$-caterpillar.

Caterpillar graphs. A *caterpillar of height at most h and length at most t*, or shortly *(h, t)-caterpillar* is a graph F constructable as follows: take a sequence of rooted trees in $\mathcal{T}_h^\bullet$, i.e., $(L_1, s_1), \ldots, (L_t, s_t)$ and connect consecutive roots with edges so that vertices $s_1, s_2, \ldots, s_t$ form a path S. We call S a *spine*, the rooted trees *legs*, and their sequence a *leg sequence*. We denote the class of all caterpillars of height at most h by $\mathcal{C}_h$, and by $\mathcal{C}_{h,t}$ the subclass of (h, t)-caterpillars.

For instance, every caterpillar in $\mathcal{C}_0$ is a path, $\mathcal{P} = \mathcal{C}_0$, see examples in Figure 6. The folklore caterpillars here correspond to $\mathcal{C}_1$ and are often used in graph theory [Harary and Schwenk, 1973, El-Basil, 1987]. Another generalization of caterpillars via *hair-length* is due to Monien [1986].

4.1 Homomorphism Counts over Caterpillars are as Expressive as Colored Walks

Let χ be a Σ-coloring on a graph G with n vertices. We define a sequence of multisets $\mathsf{wr}^{(t)}(G,\chi)$ in $\mathbb{N}^{\Sigma^t}$ for each $t \geqslant 0$ with $\boldsymbol{a}$ in Σ^t as: $\mathsf{wr}^{(0)}(G,\chi)[\lambda] = n$, $\mathsf{wr}^{(t)}(G,\chi)[\boldsymbol{a}]$ equals the number of occurrences of $\boldsymbol{a}$ in G, and $\mathsf{wr}(G,\chi) = \{\mathsf{wr}^{(t)}(G,\chi) \mid t \text{ in } \mathbb{N}\}$.

Note that our colored walk refinement is distinct from what is usually called walk refinement, i.e. [Lichter et al., 2019]. The following result explains the importance of colored walks, which is not ad hoc but rather structurally grounded.

Theorem 4.1. *For every $h, t \geqslant 0$, it holds that* $\mathsf{hom}(\mathcal{C}_{h,t}, -) \equiv \mathsf{wr}^{(t)}(-, \chi_{\mathsf{cr}}^{(h)})$.

(Proof in Appendix B). A direct consequence of Theorem 4.1 is: to capture the expressive power of homomorphism counts over folklore caterpillars (for instance) of length t, it suffices to color the vertices by their degrees and record every occurrence of a colored walk of length t by $\mathsf{wr}^{(t)}(G, \chi_{\deg})$, see Figure 1 for example of such a graph homomorphism.

4.2 Graph-Level Aggregation is as Expressive but Tractable

The previous result depicted more clearly the semantics of caterpillar homomorphisms, which is still not computationally tractable. As we observe, the number of distinct colored walks in a graph can be large, exponential in the worst case. Therefore, it is crucial that we introduce a more reduced but as expressive representation of $\mathsf{wr}(-,\chi)$, which we state for *any* coloring of vertices.

Theorem 4.2. *For every coloring χ, it holds that* $\mathsf{wr}(-,\chi) \equiv \mathcal{I}_R(-,\chi)$.

(Proof in Appendix A). Sketch of proof. Central to our arguments is that all the colored walks occurring together in a graph G with a Σ-coloring χ form a regular language. As a consequence, rather than (naively) enumerating all elements of the language, we construct an automaton $\mathcal{A}(G,\chi)$ recognizing such a language. To reflect multiplicities of occurrences in the language, we use a weighted variant of finite automata introduced by Tzeng [1996].

The "$\sqsupseteq$-direction" of the equivalence requires that $\mathcal{I}_{\mathrm{R}}(G,\chi)$ derived from $\mathcal{A}(G,\chi)$ depends purely on the recognized language and ordering on Σ via the canonical walk selection (S_t in Section 3.1). For the other "$\sqsubseteq$-direction", given two automata recognizing distinct languages, subsequent matrices of $\mathcal{I}_{\mathrm{R}}$ decompose back to automata transitions (Lemma A.13) to detect the difference.

5 Experiments

We next turn to an empirical analysis of Caterpillar GNN (Equation 4) incorporating graph-level aggregation (GLA). Because expressivity of GLA (Section 4) is controlled by its height, we propose experiments to empirically evaluate the behavior of the resulting inductive bias. Two scenarios are considered: (I.) a controlled benchmark isolating topology-driven preference for stronger task-aligned inductive bias, and (II.) real-world graph-level tasks investigating the impact of height (Section 3.3) on the trade-off between nodal efficiency and performance. We defer full implementation details to Appendix C, and the training setup to Appendix D.

5.1 Scenario I: Reducing a Bottleneck

Prior to any processing of a graph (V,E), the neighborhood topology $\tau(E)$ on V specifies which vertices are considered close, namely those in neighborhoods. We instead consider an alternative *incidence topology* $\tau(\chi)$ on V, induced by a coloring χ: two vertices are considered close if they are incident *or* adjacent to a common colored walk of length T. Since a colored walk may have multiple occurrences, this captures relationships beyond direct neighbors. We use $\tau(\chi)$ as a model to study different inductive biases in graph learning, grounded in lower-order concepts such as colored walks as shown by Theorem 4.1.

We illustrate this with our NSTEPADDITION benchmark. Given two integers of at most T bits, take a graph with two occurrences of a colored walk $a_1 \cdots a_T$. We associate each number with one occurrence as follows: encode the i-th bit of the integer in a vertex adjacent to $a_1 \cdots a_i$. This yields a graph embedding of two T-bit integers, and the classification task is whether their sum equals a target integer N. Under the incidence topology $\tau(\chi)$ corresponding bit positions are naturally close, while standard topology $\tau(E)$ may obscure such alignments. Therefore, we evaluated Caterpillar GNN on NSTEPADDITION with increasing height h, and compared to MPGNN. The results are presented in Figure 7 with more detailed information in Appendix D.

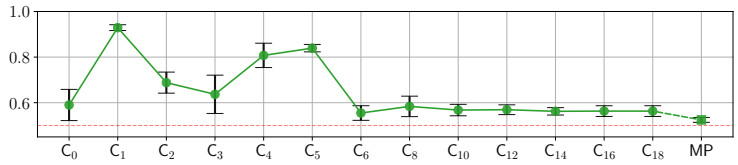

Figure 7: NSTEPADDITION: Trading off expressivity for a suitably structured task-aligned inductive bias. C_h denotes a model of Caterpillar GNN with height h (ours), while MP refers to a model of MPGNN. The y-axis shows validation accuracy.

Importance of topology. The extremal results for C_1 and MP, see Figure 7, highlight a clear difference between the two models. In MPGNN, information propagates according to $\tau(E)$. A hypothesis arises that Caterpillar GNN propagates information according to topology $\tau(\chi)$. We validate empirically on NSTEPADDITION that model C_1, as well as topology $\tau(\chi_{\deg})$, aligns bit positions for effective digit-by-digit addition, while MP within $\tau(E)$ effectively promotes learning values separately for each input pair, resulting in almost missing generalization. We detail the observed behavior in Section D.1.

5.2 Scenario II: Nodal Efficiency

We evaluate GNNs (Figure 15) on TUDataset [Morris et al., 2020] with respect to *nodal efficiency*, i.e. the average number of nodes of the computational graph (Figure 4). We fixed the number of layers for models to ensure a relative comparison. To this end, hyperparameters are per-dataset, so that the behavior can be attributed solely to the height parameter. In our experiments (Figure 15), every real-world dataset exhibits a unique behavior under increasing height. This suggests that every type of data may contain patterns organized in varying topologies resulting in distinct preferences for inductive biases. Effectively, the height parameter *shapes the model performance and nodal efficiency*. Choosing an appropriate height may improve both accuracy and nodal efficiency (Table 6).

Alongside nodal efficiency, we analyze the density of the induced computational graphs, both relative to MP and as the fraction of nonzeros in the reduced matrices (Section C.2). Empirically, for five layers, the reduced matrices have a similar or smaller number of nonzero entries than MP, depending on the height h (Table 2). However, lower height can increase absolute density: for example, MalNet-Tiny rises from 0.8% under MP to 24.7% under GLA, ogbg-molpcba from 8.8% to 22.0%, and Peptides-func from 2.1% to 15.4% (Table 4). Thus, sparse input graphs often yield smaller but denser representations, which can be favorable for dense linear-algebra kernels.

6 Related Work

Graph homomorphisms are an active area of research in graph learning. One line of work uses homomorphism counts directly as *features* [Barceló et al., 2021, Maehara and NT, 2024, Jin et al., 2024], or as *embeddings* [Nguyen and Maehara, 2020, Thiessen et al., 2022]. Several *extensions* of MPGNNs formally demonstrate the expressivity of homomorphism counts over classes extending beyond trees, listed in Table 1, including [Zhang et al., 2023a, Paolino et al., 2024]. Another line of work enhances expressivity without relating to homomorphisms. This includes cycle representations [Yan et al., 2022, Bause et al., 2025], path representations [Michel et al., 2023, Graziani et al., 2024], distance encodings [Li et al., 2020, Zhang et al., 2023b], and spectral information such as [Defferrard et al., 2017, Kreuzer et al., 2021], which is in contrast to our study of lower expressivity.

In our results, we rely on theoretical study of homomorphism counts that traces back to Lovász [1967, 2012], and their connection to Weisfeiler-Leman refinement [Weisfeiler and Leman, 1968], which is due to Dvořák [2010], Dell et al. [2018]. Further developments include quantum isomorphism via homomorphisms over planar graphs [Mančinska and Roberson, 2020], with further connection explored by Grohe et al. [2025], Kar et al. [2025]. Connections of weighted automata and homomorphism canonization given in this work motivated tight bounds for computational complexity of homomorphism indistinguishability [Černý and Seppelt, 2026], originally studied by Seppelt [2024], and for complexity of weighted automata equivalence, followed by recent improvements in [Chistikov et al., 2026]. Our canonization generalizes to higher-order colored walks [Černý, 2026].

Learning on sequential patterns such as walks has also been approached via non-equivariant random-walk kernels [Borgwardt et al., 2005, Kriege, 2022]. Other work investigates slowing down message-passing as a regularizing inductive bias [Bause and Kriege, 2022]. Recently, the role of the computational graph in deep learning has been explored [Vitvitskyi et al., 2025]. In addition, least squares-based operators have been applied to cross-network optimization [Wang et al., 2024], or graph coarsening [Jin et al., 2020, Stamm et al., 2023]. These operators target graphs at a different level of abstraction, not considering layer-specific walk incidence matrices or homomorphism counts.

7 Conclusion

We introduced a height-parameterized mechanism for controlling the computational graph of GNNs. The resulting *Caterpillar GNNs* expose a controlled trade-off: lower heights yield condensed computational graphs, while higher heights recover richer structural information. We characterize this hierarchy through colored walks and homomorphism counts over generalized caterpillars. Empirically, the mechanism can improve validation accuracy while substantially reducing computational nodes; on unattributed IMDB-BINARY, one setting retains approximately 6% of the MP baseline nodes (Table 6).

Motivating application regimes are those in which the objective of a graph-level task depends on sequential patterns representable by colored walks, while additional branching patterns are less relevant. The benchmark NStepAddition and similar tasks within the walk incidence topology realize such a regime by construction. Downstream tasks on proteins, polymers, and other caterpillar-like structures motivate targeted evaluation when their labels depend primarily on shared backbone patterns.

Reduced walk-based matrices are precomputed and reused across training epochs, and hence practical runtime depends on preprocessing, graph size, and matrix density, which we explored using our experimental implementation (Sections C.2 to C.4; Tables 4 and 7). These factors motivate efficient implementations that exploit the smaller layers of computational graphs.

The height of GLA governing the expressive power of this homomorphism canonization is currently selected by validation over a discrete range, a heuristical or even differentiable selection is a direction for future work. Further directions include directed or edge-labeled graphs and richer equivariant architectures, such as investigating analogous controls to treat the expressivity gap between 1-WL and 2-WL or Subgraph GNNs. To extend our approach from graph-level to node-level prediction, additional care is needed to preserve output vertices during the reduction process which is currently based purely on colored-walk incidences.

Acknowledgments and Disclosure of Funding

I thank Tim Seppelt, Petar Veličković, and Martin Ritzert for their insightful early feedback and valuable corrections to the initial version of the paper. I also thank Matthias Lanzinger, Diaaeldin Taha, Len Feremans, Joey De Pauw, and Jens Leysen for their valuable suggestions, careful reading, and encouragement throughout this project. I thank the anonymous reviewers for their constructive feedback and careful reading of the paper and its appendix. I am especially grateful to Floris Geerts for his sustained guidance since the inception of this project.

Marek Černý is funded by the University of Antwerp (BOF, Doctoral project 47103).

References

Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication, 2024.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

László Babai and Luděk Kučera. Canonical labelling of graphs in linear average time. In *Symposium on Foundations of Computer Science*, pages 39–46, 1979.

Pablo Barceló, Floris Geerts, Juan Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25280–25293. Curran Associates, Inc., 2021.

Franka Bause and Nils Morten Kriege. Gradual weisfeiler-leman: Slow and steady wins the race. In *The First Learning on Graphs Conference*, 2022.

Franka Bause, Fabian Jogl, Patrick Indri, Tamara Drucks, David Penz, Nils Morten Kriege, Thomas Gärtner, Pascal Welke, and Maximilian Thiessen. Maximally expressive GNNs for outerplanar graphs. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(1): 47–56, January 2005. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti1007.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. ISSN 1439-6912. doi: 10.1007/BF01305232.

Dexiong Chen, Till Hendrik Schulz, and Karsten Borgwardt. Learning long range dependencies on graphs via random walks, 2024.

Dmitry Chistikov, Radosław Piórkowski, Neha Rino, and Brink van der Merwe. Algorithms and fine-grained complexity for nondeterministic and symmetric difference automata, 2026.

Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017.

H. Dell, M. Grohe, and G. Rattan. Lovász meets Weisfeiler and Leman. In *International Colloquium on Automata, Languages, and Programming*, pages 40:1–40:14, 2018.

James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007. doi: 10.1007/s00211-007-0114-x.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Liò, and Michael Bronstein. On over-squashing in message passing neural networks: the impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 6 edition, 2025. ISBN 978-3-662-70106-5. doi: 10.1007/978-3-662-70107-2.

Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64, 2010.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Sherif El-Basil. Applications of caterpillar trees in chemistry and physics. *Journal of Mathematical Chemistry*, 1:153–174, 1987. doi: 10.1007/BF01205666.

Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and extending subgraph GNNs by rethinking their symmetries. *CoRR*, 2022.

Scott Freitas, Yuxiao Dong, Joshua Neil, and Duen Horng Chau. A large-scale database for graph representation learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Jingchu Gai, Yiheng Du, Bohang Zhang, Haggai Maron, and Liwei Wang. Homomorphism expressivity of spectral invariant graph neural networks, 2025.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org, 2017.

Caterina Graziani, Tamara Drucks, Fabian Jogl, Monica Bianchini, Franco Scarselli, and Thomas Gärtner. The expressive power of path-based graph neural networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16226–16249. PMLR, 21–27 Jul 2024.

Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations. *Advances in Combinatorics*, apr 2 2025. doi: 10.19086/aic.2025.4.

Frank Harary and Allen J. Schwenk. The number of caterpillars. *Discrete Mathematics*, 6(4):359–365, 1973. ISSN 0012-365X. doi: 10.1016/0012-365X(73)90067-8.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.

N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988*, pages 59–81, 1990.

John J. Irwin, T. Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52:1757 – 1768, 2012.

Emily Jin, Michael Bronstein, İsmail İlkan Ceylan, and Matthias Lanzinger. Homomorphism counts for graph neural networks: All about that basis, 2024.

Yu Jin, Andreas Loukas, and Joseph JaJa. Graph coarsening with preserved spectral properties. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4452–4462. PMLR, 26–28 Aug 2020.

Prem Nigam Kar, David E. Roberson, Tim Seppelt, and Peter Zeman. NPA Hierarchy for Quantum Isomorphism and Homomorphism Indistinguishability. *Leibniz Int. Proc. Inf.*, 334:105:1–105:19, 2025. doi: 10.4230/LIPIcs.ICALP.2025.105.

Stefan Kiefer. Notes on equivalence and minimization of weighted automata. *CoRR*, abs/2009.01217, 2020.

Stefan Kiefer, Andrzej Murawski, Joel Ouaknine, Bjoern Wachter, and James Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *Logical Methods in Computer Science*, Volume 9, Issue 1:8, Mar 2013. ISSN 1860-5974. doi: 10.2168/LMCS-9(1:8)2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Nils M. Kriege. Weisfeiler and leman go walking: Random walk kernels revisited. In *Advances in Neural Information Processing Systems*, volume 35, pages 20119–20132. Curran Associates, Inc., 2022.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: design provably more powerful neural networks for graph representation learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

M. Lichter, I. Ponomarenko, and P. Schweitzer. Walk refinement, walk logic, and the iteration number of the Weisfeiler-Leman algorithm. In *Symposium on Logic in Computer Science*, pages 1–13, 2019.

L. Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3): 321–328, sep 1967. ISSN 1588-2632. doi: 10.1007/BF02280291.

László Miklós Lovász. Large networks and graph limits. In *Colloquium Publications*, 2012.

László Lovász and Balázs Szegedy. Contractors and connectors of graph algebras. *Journal of Graph Theory*, 60(1):11–30, 2009. doi: 10.1002/jgt.20343.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic GNNs be strong baselines for graph-level tasks? simple architectures meet excellence. In *Forty-second International Conference on Machine Learning*, 2025.

Takanori Maehara and Hoang NT. Deep homomorphism networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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.

Gaspard Michel, Giannis Nikolentzos, Johannes Lutzeyer, and Michalis Vazirgiannis. Path neural networks: expressive and accurate graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Burkhard Monien. The bandwidth minimization problem for caterpillars with hair length 3 is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 7(4):505–512, 1986. doi: 10.1137/0607057.

C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *CoRR*, 2020.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, pages 4602–4609, 2019.

Daniel Neuen and Tim Seppelt. Distinguishing graphs by counting homomorphisms from sparse graphs, 2026.

Hoang Nguyen and Takanori Maehara. Graph homomorphism convolution. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7306–7316. PMLR, 13–18 Jul 2020.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

Raffaele Paolino, Sohir Maskey, Pascal Welke, and Gitta Kutyniok. Weisfeiler and leman go loopy: A new hierarchy for graph representational learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 120780–120831. Curran Associates, Inc., 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Andrzej Proskurowski and Jan Arne Telle. Classes of graphs with restricted interval models. *Discrete Mathematics and Theoretical Computer Science*, Vol. 3 no. 4(4):167–176, January 1999. doi: 10.46298/dmtcs.263.

C. Qian, G. Rattan, F. Geerts, C. Morris, and M. Niepert. Ordered subgraph aggregation networks. In *Advances in Neural Information Processing Systems*, 2022.

Gaurav Rattan and Tim Seppelt. Weisfeiler-leman, graph spectra, and random walks. *ArXiv preprint*, 2021.

David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree, 2022.

Georg Schindling. Homomorphism indistinguishability and game comonads for restricted conjunction and requantification, 2025.

M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2): 245–270, 1961. ISSN 0019-9958. doi: 10.1016/S0019-9958(61)80020-X.

Tim Seppelt. An algorithmic meta theorem for homomorphism indistinguishability. In *49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024)*, volume 306 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:19, 2024. ISBN 978-3-95977-335-5. doi: 10.4230/LIPIcs.MFCS.2024.82.

Felix I. Stamm, Michael Scholkemper, Markus Strohmaier, and Michael T. Schaub. Neighborhood structure configuration models. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 210–220, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583266.

Maximilian Thiessen, Pascal Welke, and Thomas Gärtner. Expectation complete graph representations using graph homomorphisms. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Wen-Guey Tzeng. On path equivalence of nondeterministic finite automata. *Inf. Process. Lett.*, 58(1): 43–46, April 1996. ISSN 0020-0190.

Marek Černý. On numbers of simplicial walks and equivalent canonizations for graph recognition, 2026.

Marek Černý and Tim Seppelt. Homomorphism indistinguishability, multiplicity automata equivalence, and polynomial identity testing. In *43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026)*, volume 364 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:20, 2026. ISBN 978-3-95977-412-3. doi: 10.4230/LIPIcs.STACS.2026.25.

Alex Vitvitskyi, João G. M. Araújo, Marc Lackenby, and Petar Veličković. What makes a good feedforward computational graph?, 2025.

Junya Wang, Yi-Jiao Zhang, Cong Xu, Jiaze Li, Jiachen Sun, Jiarong Xie, Ling Feng, Tianshou Zhou, and Yanqing Hu. Reconstructing the evolution history of networked complex systems. *Nature Communications*, 15(1):2849, 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-47248-x.

Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 1–11, 1973. doi: 10.1109/SWAT.1973.13.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*, 1968.

Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, pages 513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Cycle representation learning for inductive relation prediction. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

Dingyi Zeng, Wenyu Chen, Wanlong Liu, Li Zhou, and Hong Qu. Rethinking random walk in graph representation learning. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096316.

Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph GNNs via subgraph weisfeiler-lehman tests. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 41019–41077. PMLR, 23–29 Jul 2023a.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023b.

Bohang Zhang, Lingxiao Zhao, and Haggai Maron. On the expressive power of spectral invariant graph neural networks, 2024.

Table of Contents

A Weighted Automata **17**

A.1 Graph Walks and Automata Semantics 18

A.2 Minimization of Weighted Automata 21

A.3 Constraints on Walk-Incidence Matrices 24

B Homomorphism Counts and Colored Walks **27**

B.1 Quantum Graph Homomorphisms 30

B.2 Strict separation 33

B.3 Expressivity hierarchy 34

C Graph Neural Networks **34**

C.1 Caterpillar GCN 34

C.2 Density of Computational Graphs 35

C.3 Impacts of Relative Density on Asymptotic Complexity 36

C.4 Towards Modern Architectures and Large Graph-Level Datasets 37

D Details of Experiments **38**

D.1 Additional Conclusions for Paradoxically Reversed Descent 38

D.2 Incidence Topology 39

D.3 Generation of nStepAddition Dataset 39

D.4 Implementation Details 40

D.5 Experimental Setting on Real-World Datasets 41

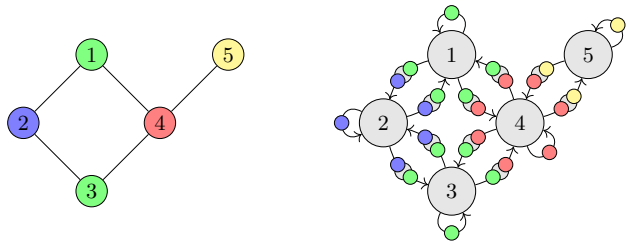

Figure 8: (a) An example graph G on vertices $\{1, 2, 3, 4, 5\}$ with a Σ-coloring χ, where $\Sigma = \{\mathsf{r}, \mathsf{b}, \mathsf{g}, \mathsf{y}\}$. (b) The weighted automaton $\mathcal{A}(G, \chi)$ that accepts a weighted language of words corresponding to colored walks in G (Lemma A.5). The weights of the language represent the number of occurrences of each walk in G. Transitions corresponding to directed edges uv in $E(G)$ are represented by matrices $\boldsymbol{M}(\chi(u)|\chi(v)) = \boldsymbol{P}_{\chi(u)}\boldsymbol{A}\boldsymbol{P}_{\chi(v)}$, while transitions corresponding to loops at vertex u in $V(G)$ are given by matrices $\boldsymbol{M}(\chi(u)) = \boldsymbol{P}_{\chi(u)}\boldsymbol{I}\boldsymbol{P}_{\chi(u)}$.

A Weighted Automata

In this section, we introduce *weighted automata* [Schützenberger, 1961, Tzeng, 1996] alternatively called *multiplicity automata* in the theoretically oriented literature [Kiefer et al., 2013]. Then, we apply insights from automata theory, using them as a key technical tool to establish the results of Section 3 and Theorem 4.2 from Section 4.

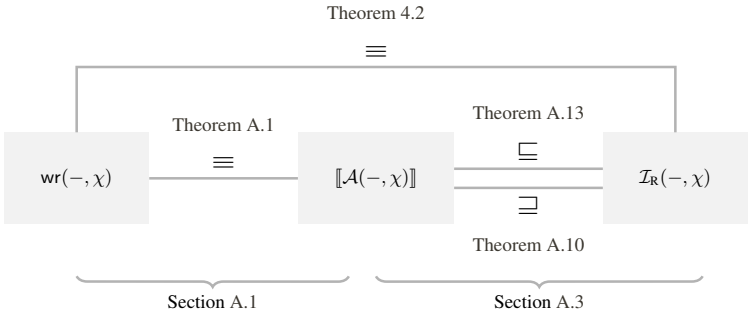

Figure 9: Sections A.1 and A.3 establish the result of Theorem 4.2 for a coloring χ by relating the following three functions on graphs: Colored walk refinement $\mathsf{wr}(-, \chi)$, semantics of graph walk automaton $[\![\mathcal{A}(-, \chi)]\!]$, and the matrices of graph-level aggregation $\mathcal{I}_{\mathsf{R}}(-, \chi)$.

Automata. A *weighted finite automaton*, or here simply *automaton*, is a tuple

$$\mathcal{A} = (Q, \Sigma, \boldsymbol{M}, \alpha, \omega), \tag{5}$$

where Q is a finite set of states; Σ is a finite alphabet; $\boldsymbol{M}(-)\colon \Sigma \to \mathbb{R}^{Q\times Q}$ is a per-symbol mapping of transition matrices, α in $\mathbb{R}^{1\times Q}$ is the initial state (row) vector, and ω in $\mathbb{R}^{Q\times 1}$ is the final state (column) vector.

Semantics. Given an automaton $\mathcal{A} = (Q, \Sigma, \boldsymbol{M}, \alpha, \omega)$, we extend the mapping $\boldsymbol{M}\colon \Sigma \to \mathbb{R}^{Q\times Q}$ to words as follows: for a given word $\boldsymbol{w} = w_1 w_2 \dots w_t$ in Σ^*, we define $\boldsymbol{M}(\boldsymbol{w}) = \boldsymbol{M}(w_1)\boldsymbol{M}(w_2)\cdots\boldsymbol{M}(w_t) \in \mathbb{R}^{Q\times Q}$, and $\boldsymbol{M}(\lambda) = I \in \mathbb{R}^{Q\times Q}$ for the empty word λ in Σ. The *semantics* of the automaton $\mathcal{A}$ is a function $[\![\mathcal{A}]\!]\colon \Sigma^* \to \mathbb{R}$, interpreted as a formal series, defined by

$$[\![\mathcal{A}]\!](\boldsymbol{w}) = \alpha\boldsymbol{M}(\boldsymbol{w})\omega \in \mathbb{R} \quad \text{for all } \boldsymbol{w} \text{ in } \Sigma^*. \tag{6}$$

Two automata $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if their semantics are equivalent, that is, $[\![\mathcal{A}]\!](\boldsymbol{w}) = [\![\mathcal{A}']\!](\boldsymbol{w})$ for all $\boldsymbol{w}$ in Σ^*. The value $[\![\mathcal{A}]\!](\boldsymbol{w})$ is called the *weight* of $\boldsymbol{w}$. For a symbol a in Σ, let a^k denote the word formed by repeating a exactly k-times.

A.1 Graph Walks and Automata Semantics

As in the main text, we assume a graph G and χ a Σ-coloring on G, for which we now define a weighted finite automaton $\mathcal{A}(G,\chi) = (Q, \overline{\Sigma}, \boldsymbol{M}, \mathbf{1}^\top, \mathbf{1})$ defined as follows. The states are corresponding to the vertices of the graph $Q \coloneqq V(G)$, and the alphabet is induced by the colors of the vertices and edges as follows:

$$\overline{\Sigma} \coloneqq \{\chi(u) \mid u \in V(G)\} \cup \{\chi(u)|\chi(v) \mid uv \in E(G)\}, \tag{7}$$

where we consider both a and $a|b$ as a single symbol in $\overline{\Sigma}$, for some original colors a, b in Σ. For a in $\overline{\Sigma}$, the *partition matrix* $\boldsymbol{P}_a$ in $\mathbb{R}^{V\times V}$ is the diagonal matrix defined as $\boldsymbol{P}_a[u,u] = 1$ if $\chi(u) = a$ and 0 otherwise. For each symbol a or $a|b$ in $\overline{\Sigma}$, we define the transition matrices using the adjacency matrix of G and the coloring-dependent partition matrices as

$$\boldsymbol{M}(a) \coloneqq \boldsymbol{P}_a = \boldsymbol{P}_a \boldsymbol{I} \boldsymbol{P}_a \text{ in } \mathbb{R}^{V\times V} \text{ and } \boldsymbol{M}(a|b) \coloneqq \boldsymbol{P}_a \boldsymbol{A} \boldsymbol{P}_b \text{ in } \mathbb{R}^{V\times V}.$$

We set the initial and final vectors to the all-one vectors. We depict an example of a graph and a coloring and the corresponding automata in Figure 8.

We also recall the following from the main paper. For a given graph G with χ a Σ-coloring, and a given length $t \geqslant 0$, we define the *walk-incidence matrix* $\boldsymbol{W}_t = \boldsymbol{W}_t(G,\chi)$ of shape $V \times \Sigma^t$ for each u in V and $\boldsymbol{a}$ in Σ^t by

$$\boldsymbol{W}_t[u, \boldsymbol{a}] \text{ is the number of occurrences of } \boldsymbol{a} \text{ that terminate in vertex } u. \tag{8}$$

Colored walk refinement. Let χ be a Σ-coloring on a graph G with n vertices. A *colored walk refinement* is a sequence of multisets $\mathsf{wr}^{(t)}(G,\chi)$ in $\mathbb{N}^{\Sigma^t}$ defined for each $t \geqslant 0$ with $\boldsymbol{a}$ in Σ^t as

$$\begin{aligned}
&\mathsf{wr}^{(0)}(G,\chi)[\lambda] = n,\\
&\mathsf{wr}^{(t)}(G,\chi)[\boldsymbol{a}] \quad \text{equals the number of occurrences of } \boldsymbol{a} \text{ in G, and}\\
&\qquad \mathsf{wr}(G,\chi) = \{\mathsf{wr}^{(t)}(G,\chi) \mid t \text{ in } \mathbb{N}\}.
\end{aligned}$$

The main result of this subsection is the equivalence of graph-induced weighted finite automata and colored walk refinement.

Theorem A.1. *For every coloring χ it holds that*

$$[\![\mathcal{A}(-,\chi)]\!] \equiv \mathsf{wr}(-,\chi).$$

An implication of Theorem A.1 is that instead of the sums of columns of walk-incidence matrices, we can walk directly with the automata semantics, for which we can *choose* the representation of the automaton potentially more suitable for our application.

To prove this theorem, we need a couple of intermediate results. We first show that the partition matrices $\boldsymbol{P}_a$ are projection matrices.

Proposition A.2 (Partition matrices)**.** *Let a and b be colors in Σ then for the partition matrices the following hold:*

1. *$\boldsymbol{P}_a = \boldsymbol{P}_a^2$ is a projection, and*
2. *$\boldsymbol{P}_a \boldsymbol{P}_b$ is all-zero if $a \neq b$.*

Proof. For the first part, we have for all u, v in $V(G)$:

$$(\boldsymbol{P}_a \boldsymbol{P}_a)[u,v] = \sum_{w\in V(G)} \boldsymbol{P}_a[u,w]\boldsymbol{P}_a[w,v] = \sum_{w=u} 1 \cdot \boldsymbol{P}_a[w,v] = \sum_{u=w=v} 1 \cdot 1,$$

which is 1 if $u = v$ and 0 otherwise. Similarly, for the second part, given the colors a, b in Σ we have for all u, v in $V(G)$:

$$(\boldsymbol{P}_a \boldsymbol{P}_b)[u,v] = \sum_{w\in V(G)} \boldsymbol{P}_a[u,w]\boldsymbol{P}_b[w,v] = \sum_{\substack{u=w\\ a=\chi(w)}} 1 \cdot \boldsymbol{P}_b[w,v] = \sum_{\substack{u=w, w=v\\ a=\chi(w), \chi(w)=b}} 1 \cdot 1,$$

which is 1 if $u = v$ and $a = b$, and 0 otherwise. Thus, $\boldsymbol{P}_a \boldsymbol{P}_b$ is all-zero matrix if $a \neq b$. □

Observation A.3. *Let $\boldsymbol{w}$ in $\overline{\Sigma}$ be a word with a non-zero weight, $[\![\mathcal{A}(G,\chi)]\!](\boldsymbol{w}) > 0$, then $\boldsymbol{w}$ is of the following form:*

$$(a_1^{k_1})(a_1|a_2)(a_2^{k_2})(a_2|a_3)(a_3^{k_3})\cdots(a_{t-1}^{k_{t-1}})(a_{t-1}^{k_{t-1}}|a_t^{k_t})(a_t^{k_t}),$$

where a_i in Σ are colors, k_i in $\mathbb{N}$ are non-negative integers,

Proof. For the sake of contradiction, consider a word $\boldsymbol{w}$ that contains a subword $ab|c$ in $\overline{\Sigma}$, $c|ba$ in $\overline{\Sigma}$, or ab in $\overline{\Sigma}$ such that $a \neq b$ and weight $[\![\mathcal{A}(G,\chi)]\!](\boldsymbol{w}) > 0$. In the first case, by the definition of semantics, we get for any $\boldsymbol{w}_1, \boldsymbol{w}_2$ in Σ^*:

$$\begin{aligned}[\![\mathcal{A}(G,\chi)]\!](\boldsymbol{w}_1 ab|c\, \boldsymbol{w}_2) &= \mathbf{1}^\top \boldsymbol{M}(\boldsymbol{w}_1)\boldsymbol{M}(a)\boldsymbol{M}(b|c)\boldsymbol{M}(\boldsymbol{w}_2)\mathbf{1}\\ &= \mathbf{1}^\top \boldsymbol{M}(\boldsymbol{w}_1)\boldsymbol{P}_a\boldsymbol{P}_b\boldsymbol{A}\boldsymbol{P}_c\boldsymbol{M}(\boldsymbol{w}_2)\mathbf{1}\\ &= \mathbf{1}^\top \boldsymbol{M}(\boldsymbol{w}_1)0\boldsymbol{M}(\boldsymbol{w}_2)\mathbf{1} = 0,\end{aligned}$$

where we used (1.) of Proposition A.2 for the second equality. Thus, we have $[\![\mathcal{A}(G,\chi)]\!](\boldsymbol{w}_1 ab|c\, \boldsymbol{w}_2) = 0$, and symmetrically $[\![\mathcal{A}(G,\chi)]\!](\boldsymbol{w}_1 c|ba\, \boldsymbol{w}_2) = 0$, and analogously $[\![\mathcal{A}(G,\chi)]\!](\boldsymbol{w}_1 ab\, \boldsymbol{w}_2) = 0$. □

Lemma A.4. *For all t in $\mathbb{N}$, a_i in Σ and k_i in $\mathbb{N}$ for i in $[t]$ the weight of the following words is equal:*

1. $(a_1^{k_1})(a_1|a_2)(a_2^{k_2})(a_2|a_3)(a_3^{k_3})\cdots(a_{t-1}^{k_{t-1}})(a_{t-1}^{k_{t-1}}|a_t^{k_t})(a_t^{k_t})$, *and*
2. $(a_1|a_2)(a_2|a_3)\cdots(a_{t-1}|a_t)$.

Proof. This follows from (1) of Proposition A.2. Indeed, it suffices to observe that

$$\begin{aligned}&\boldsymbol{M}\left(a_i^{k_i}\right) = \boldsymbol{M}\left(a_i\right), \text{ and}\\ &\boldsymbol{M}\left((a_i^{k_i})(a_i|a_{i+1})\,(a_{i+1}^{k_{i+1}})\right) = \boldsymbol{P}_{a_i}^{k_i}\boldsymbol{P}_{a_i}\boldsymbol{A}\boldsymbol{P}_{a_{i+1}}\boldsymbol{P}_{a_{i+1}}^{k_{i+1}} = \boldsymbol{P}_{a_i}\boldsymbol{A}\boldsymbol{P}_{a_{i+1}} = \boldsymbol{M}\left(a_i|a_{i+1}\right).\end{aligned}$$

Then, from the semantics of automata, the statement on the equal weights follows. □

Therefore, by Lemma A.4, we can characterize the semantics of automata $\mathcal{A}(G,\chi)$, using only words of the form as in Item 2 without loss of generality. Consequently, by Observation A.3, we shall use a more natural *simplified notation*

$$a_1|a_2|a_3\cdots a_{t-1}|a_t$$

for the word $(a_1|a_2)(a_2|a_3)\cdots(a_{t-1}|a_t)$.

We now make the first connection between weighted automata and walk-incidence matrices.

Lemma A.5. *Let G be a graph, χ a Σ-coloring on G, and $(w_1, w_2, \ldots, w_t)$ in Σ^t a colored walk. Then the following holds:*

$$\boldsymbol{M}\left(w_t|w_{t-1}|\ldots|w_1\right)\mathbf{1} = \boldsymbol{W}_t(G,\chi)[-, w_1w_2\cdots w_t]. \tag{9}$$

Proof. By induction on the length t. For the base case $t = 0$, it holds $\boldsymbol{M}(\lambda)\mathbf{1} = \boldsymbol{I}\mathbf{1} = \mathbf{1} = \boldsymbol{W}_0(G,\chi)[-,\lambda]$.

For the induction step, suppose that the lemma holds for t. We take a word $w_1|w_2|\ldots|w_{t+1}$ and consider a vertex u in $V(G)$. Then,

$$\begin{aligned}
(\boldsymbol{M}\left(w_{t+1}|w_t|\ldots|w_1\right)\mathbf{1})[u] &= (\boldsymbol{M}(w_{t+1}|w_t)\boldsymbol{M}\left(w_t|w_{t-1}|\ldots|w_1\right)\mathbf{1})\left[u\right] \\
&= \sum_{v\in V(G)} \boldsymbol{M}(w_{t+1}|w_t)\boldsymbol{W}_t(G,\chi)[v, w_1w_2\cdots w_t] \\
&= \sum_{v\in V(G)} \boldsymbol{P}_{w_{t+1}}[u,u]\boldsymbol{A}[u,v]\boldsymbol{P}_{w_t}[v,v]\boldsymbol{W}_t(G,\chi)[v, w_1w_2\cdots w_t] \\
&= \sum_{\substack{v\in V(G)\\ w_{t+1}=\chi(v)\\ w_t=\chi(u)}} \boldsymbol{A}[v,u]\boldsymbol{W}_t(G,\chi)[v, w_1w_2\cdots w_t] \\
&= \sum_{\substack{uv\in E(G)\\ \chi(v)=w_t\\ \chi(u)=w_{t+1}}} \boldsymbol{W}_t(G,\chi)[u, w_1w_2\cdots w_t] \\
&= \boldsymbol{W}_{t+1}(G,\chi)[u, w_1w_2\cdots w_{t+1}],
\end{aligned}$$

where the first and third equalities follow from the definition of $\boldsymbol{M}$, the second from the induction hypothesis, the fourth and fifth follow from the definition of $\boldsymbol{P}_{w_{t+1}}$ and A. Finally, the last equation follows from the observation that we can find every occurrence of the walk of color $w_1w_2\cdots w_tw_{t+1}$ ending in the vertex u, given that $\chi(u) = w_{t+1}$, as an occurrence of the walk of color $w_1w_2\cdots w_t$ ending at one of its neighbors v, which is of color $\chi(v) = w_t$. □

Theorem A.6. *Let G be a graph, χ a Σ-coloring on G, and $(w_1, w_2, \ldots, w_t)$ in Σ^t a colored walk. Then the following holds:*

$$[\![\mathcal{A}(G,\chi)]\!]\left(w_t|w_{t-1}|\ldots|w_1\right) = \sum_{u\in V(G)} \boldsymbol{W}_t(G,\chi)[u, w_1w_2\cdots w_t]. \tag{10}$$

Proof. By the definition of semantics, we have:

$$\begin{aligned}
[\![\mathcal{A}(G,\chi)]\!]\left(w_t|w_{t-1}|\ldots|w_1\right) &= \mathbf{1}^\top\boldsymbol{M}(w_1|w_2|\ldots|w_t)\mathbf{1} \\
&= \mathbf{1}^\top\left(\boldsymbol{M}(w_1|w_2|\ldots|w_t)\right)^\top\mathbf{1} \\
&= \mathbf{1}^\top\boldsymbol{M}(w_t|w_{t-1}|\ldots|w_1)\mathbf{1} \\
&= \mathbf{1}^\top\boldsymbol{W}_t(G,\chi)[u, w_1w_2\cdots w_t] \\
&= \sum_{u\in V(G)} \boldsymbol{W}_t(G,\chi)[u, w_1w_2\cdots w_t],
\end{aligned}$$

where the second equality follows from the symmetry of all matrices $\boldsymbol{M}$, since $\boldsymbol{P}_a$ and $\boldsymbol{A}$ are symmetric matrices, $a \in \Sigma$, the fourth equality follows from Lemma A.5. □

Proof of Theorem A.1. Follows from Theorem A.6, by definition of colored walk refinement. □

Algorithm 1: Layered canonical word search

Input: Number of layers T, ordered alphabet $(\Sigma, \leqslant)$, graph G, coloring χ
Output: $S_0, S_1, \dots S_T$ finite subsets of Σ^*

for $t \leftarrow 0$ ***to*** T **do**
  $S_t \leftarrow \varnothing$
  $\mathcal{B}_t \leftarrow \varnothing$
  queue $Q_t \leftarrow []$
$Q_0.push(\lambda)$
for $t \leftarrow 0$ ***to*** T **do**
  while ***not*** $Q_t.empty()$ **do**
    $\boldsymbol{w} \leftarrow Q_t.pop()$
    $\gamma \leftarrow \boldsymbol{W}_t(G, \chi)[-, \boldsymbol{w}]$
    if $\operatorname{rank}(\mathcal{B}_t \cup \{\gamma\}) > |\mathcal{B}_t|$ **then**
      $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{\gamma\}$
      $S_t \leftarrow S_t \cup \{\boldsymbol{w}\}$
      if $t < T$ **then**
        foreach a ***in*** Σ **do**
          $Q_{t+1}.push(\boldsymbol{w}a)$
return $S_0, S_1, \dots, S_T$

A.2 Minimization of Weighted Automata

In this subsection, we recall the definition of the *minimal weighted automaton* (cf. Kiefer et al. [2013], Kiefer [2020]). Unlike for the standard finite automata, in the weighted case, the concrete variant is unique only up to the change of the basis of the vector space $\mathbb{R}^Q$. On the bright side, every such minimal weighted automaton has the unique dimension, that is the number of states $|Q|$, and the unique (canonical) word subset $S \subseteq \Sigma^*$ of size at most $|Q|$. Also, the minimal weighted automaton can be computed in time $\mathcal{O}(n^3|\Sigma|)$.

In our case, the graph-induced automaton $\mathcal{A}(G, \chi)$ are completely symmetric in the sense that $[\![\mathcal{A}(G, \chi)]\!](w_1|w_2|\dots|w_t) = [\![\mathcal{A}(G, \chi)]\!](w_t|w_{t-1}|\cdots|w_1)$ for w_i in Σ, but also in the sense that the initial vector can be interchanged as $\alpha^\top = \omega = \mathbf{1}$, and similarly matrices $\boldsymbol{M}(a) = \boldsymbol{M}(a)^\top$ and $\boldsymbol{M}(a|b) = \boldsymbol{M}(b|a)^\top$. It follows that the forward and backward steps of the automata minimization described are spanning the same vector space. And thus if there is a minimal automaton $A_1 = (Q_1, \overline{\Sigma}, \boldsymbol{M}_1, \mathbf{1}^\top, \mathbf{1})$ for the graph induced automaton $\mathcal{A}(G, \chi)$, it holds that there is a matrix of a full rank $\boldsymbol{F}$ in $\mathbb{R}^{V \times Q_1}$ such that for all a in Σ:

$$\mathbf{1}^\top \boldsymbol{F} = \mathbf{1}^\top, \quad \mathbf{1} = \boldsymbol{F}\mathbf{1}, \quad \boldsymbol{F}\boldsymbol{M}_1(a) = \boldsymbol{M}(a)\boldsymbol{F}, \quad \boldsymbol{F}\boldsymbol{M}_1(a|b) = \boldsymbol{M}(a|b)\boldsymbol{F}. \tag{11}$$

In general, by e.g. Kiefer [2020], if we have two minimal automata $\mathcal{A}_i = (Q_i, \overline{\Sigma}, \boldsymbol{M}_i, \alpha_i, \omega_i)$ for $i = 1, 2$, then there is an invertible matrix $\boldsymbol{Q}$ in $\mathbb{R}^{Q_2 \times Q_1}$ such that for all a in Σ:

$$\alpha_2 \boldsymbol{Q} = \alpha_1, \quad \omega_2 = \boldsymbol{Q}\omega_1, \quad \boldsymbol{Q}\boldsymbol{M}_2(a) = \boldsymbol{M}_1(a)\boldsymbol{Q}, \quad \boldsymbol{Q}\boldsymbol{M}_2(a|b) = \boldsymbol{M}_1(a|b)\boldsymbol{Q}. \tag{12}$$

Inspired by the minimization procedure of weighted automata, also cf. Kiefer et al. [2013], we propose a similar procedure *layered canonical word search*, see Algorithm 1, to compute the canonical word subsets $S_t(G, \chi)$ for all t in $\mathbb{N}$, as given in Section 3.1. The main distinction from the automata minimization algorithm is that we keep separate queue, base, and the word subset for each layer t, and thus we are ensuring linear independence for each layer t separately.

We recall the Conditions given in Section 3.1: The definition proceeds inductively: $S_0(G, \chi) = \{\lambda\} = \Sigma^0$, and for known $S_t(G, \chi)$, the set $S_{t+1}(G, \chi) \subseteq \Sigma^{t+1}(G, \chi)$ satisfies the following:

(i) for every $\boldsymbol{a}c$ in $S_{t+1}(G, \chi)$ there is $\boldsymbol{a}$ in $S_t(G, \chi)$ (*prefix-closedness*),
(ii) the columns of $\boldsymbol{W}_{t+1}$ induced by $S_{t+1}(G, \chi)$ are *linearly independent*, and
(iii) the set $S_{t+1}(G, \chi)$ is *lexicographically minimal* among other sets satisfying (i) and (ii).

Lemma A.7 (Correctness)**.** *Let G be a graph, χ a Σ-coloring on G then the Algorithm 1 computes the canonical word subsets $S_t(G, \chi)$, satisfying Conditions (i), (ii), and (iii), for all t* in $\mathbb{N}$.

Proof. We proceed by induction on t. It is observed that in the for-loop of the algorithm ranging over t, we only add to the list S_t and to the set $\mathcal{B}_t$, working only with the elements from the queue Q_t, and adding new elements to the queue Q_{t+1} based on the words we added to S_t. For the base case $t = 0$, the queue Q_0 is initialized with the empty word λ, for which $\boldsymbol{W}_0[-, \lambda]$ is the all-one vector $\mathbf{1}$, and as the base $\mathcal{B}_0$ is empty and its rank 0. Thus, we have $\mathcal{B}_0 = \{\mathbf{1}\}$ and $S_0 = \{\lambda\}$.

For the induction step, we assume that the algorithm computes S_t and $\mathcal{B}_t$ correctly. In the beginning of the $(t + 1)$-th iteration, the queue Q_{t+1} contains the words of length $t + 1$, of the form $\boldsymbol{w}a$ for all a in Σ and $\boldsymbol{w}$ in S_t, thus satisfying the condition (i). If we add $\gamma = \boldsymbol{W}_{t+1}[-, \boldsymbol{w}a]$ to $\mathcal{B}_{t+1}$, and $\boldsymbol{w}$ to S_{t+1}, then we have $\operatorname{rank}(\mathcal{B}_{t+1} \cup \{\gamma\}) > |\mathcal{B}_{t+1}|$, thus words in S_{t+1} are satisfying condition (ii). The last condition (iii) follows from the fact that we always add to Q_{t+1} possible candidates by the foreach-loop over Σ in a lexicographical order, and we keep the order while processing the queue. □

Lemma A.8 (Complexity)**.** *There is an implementation of Algorithm 1 that for a given T* in $\mathbb{N}$*, computes the canonical word subsets $S_1, S_2, \ldots, S_T$, in time $\mathcal{O}(Tn^\omega)$, where n is the number of vertices in G.*

Proof. 1. Quartic bound. We focus on the cost of an iteration of the for-loop over t, the base case $t = 0$ is trivially $\mathcal{O}(|\Sigma| + n)$. The size of the queue Q_{t+1} is exactly $|S_t||\Sigma| \leqslant n|\Sigma|$. For every word $\boldsymbol{w}a$ in Q_{t+1}, we compute the corresponding vector of walk incidence matrix $\boldsymbol{W}_t[-, \boldsymbol{w}a]$ in $\mathbb{R}^{V(G)}$, from the vector $\boldsymbol{W}_t[-, \boldsymbol{w}]$ for the word $\boldsymbol{w}$ in S_t as shown in the proof of Lemma A.5, by multiplying by matrix $\boldsymbol{P}_a$ if $t = 1$, and by $\boldsymbol{P}_a\boldsymbol{A}$ if $t \geqslant 2$. The expensive part of the algorithm is the computation of the rank of $\mathcal{B}_{t+1} \cup \{\gamma\}$, this can be done in $\mathcal{O}(n^2)$ time, if we maintain the representation of the linearly independent vectors of $\mathcal{B}_t$ in a row echelon form. Thus, for a limit T in $\mathbb{N}$, we have T iterations of the for-loop over t, which has the complexity of $\mathcal{O}(|S_t||\Sigma| + |S_t|n^2 + |S_{t+1}||\Sigma|) = \mathcal{O}(n^3|\Sigma|)$. And finally, the total complexity of the algorithm is $\mathcal{O}(Tn^3|\Sigma|)$. This yields $\mathcal{O}(Tn^4)$ using that $|\Sigma| \leqslant n$.

2. Cubic bound. To improve upon the above quartic bound, we further exploit the structure of our transition matrices $\boldsymbol{P}_a\boldsymbol{A}$ and $\boldsymbol{P}_a\boldsymbol{I}$. In each iteration over t Partition matrices on the left side of $\boldsymbol{A}$ effectively zero out rows of vertices that are not of color a. Therefore, instead of performing for a given base $\boldsymbol{B}_t$ multiplication $\boldsymbol{P}_a\boldsymbol{A}\boldsymbol{B}_t$ for all a in Σ, we only multiply by $\boldsymbol{A}\boldsymbol{B}_t$ and then choose the row-submatrix $\boldsymbol{A}_a$ of shape $V_a \times V$ where $V_a = \{u \mid u \in V,\ \chi(u) = a\}$. To that end, for each $\boldsymbol{A}_a$ we select the first subset of independent columns for S_t. This is without loss of generality since the row partition $(V_{a_1} \sqcup \cdots \sqcup V_{a_{|\Sigma|}} = V)$ gives orthogonal decomposition of the column space. Using the row echelon form technique as above, this accounts for $\mathcal{O}(n|V_a|^2)$ steps.

In total, we sum $\sum_a n|V_a|^2$ over all colors in Σ, and since $n \sum_a |V_a|^2 \leqslant n \cdot n^2$, we get an algorithm of complexity $\mathcal{O}(Tn^3)$.

3. QR-bound. To obtain this bound, we focus on the linear dependence queries that cost $\mathcal{O}(|V_a|^2)$ each. Instead of running a greedy loop over columns for each $\boldsymbol{A}_a$, we use matrix decomposition.

Recall a general **QR-decomposition** of square and/or tall matrix $\boldsymbol{X}$ of shape $k \times l$ such that $k \geqslant l$. It computes a square matrix $\boldsymbol{Q}$ of shape $k \times k$ and an upper-triangular matrix $\boldsymbol{R}$ of shape $k \times l$ such that $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$. Diagonal δ of matrix $\boldsymbol{R}$ is a vector of length l, where $\delta[j] = \boldsymbol{R}[j, j]$ is zero if and only if the j-th column of $\boldsymbol{X}$ is linearly dependent on the previous columns.

It remains to clarify usage of QR-decomposition in our case. Matrices $\boldsymbol{A}_a$ of shape $V_a \times V$ such that $|V_a| \leqslant n$, are rather wide than tall. Therefore, we instead apply the following iterative approach:

Initially, duplicate rows of $\boldsymbol{A}_a$ to obtain matrix $\boldsymbol{A}_a^{(0)}$ of shape $V_a^{(0)} \times V$, where $V_a^{(0)} = V_a \uplus V_a$. Next, split columns of $\boldsymbol{A}_a^{(0)}$ into $m_0 \coloneqq \lceil n/|V_a^{(0)}| \rceil$ square blocks $\boldsymbol{A}_{a,1}^{(0)}, \ldots, \boldsymbol{A}_{a,m_0}^{(0)}$, padding the last one with zero columns if needed.

In the s-th step, compute QR-decomposition for each square matrix $\boldsymbol{A}_{a,i}^{(s)}$ independently, selecting its linearly independent columns. Crucially, due to the initial row duplication, $\operatorname{rank}(\boldsymbol{A}_{a,i}^{(s)}) \leqslant \frac{1}{2}|V_a^{(s)}|$, hence obtaining at most $\frac{1}{2}|V_a^{(s)}|$ of independent columns. From consecutive blocks $\boldsymbol{A}_{a,2i'}^{(s)}$ and

$\boldsymbol{A}^{(s)}_{a,2i'+1}$, construct one square block $\boldsymbol{A}^{(s+1)}_{a,i'}$ containing at most $2 \cdot \frac{1}{2}|V^{(s)}_a|$ selected independent columns (padding rest with zeros if needed) of $\boldsymbol{A}^{(s)}_{a,2i'}$ and $\boldsymbol{A}^{(s)}_{a,2i'+1}$. This way, we obtain $m_{s+1} := \lceil \frac{m_s}{2} \rceil$ blocks with rows remaining duplicated for the $(s+1)$-step.

Since the number of blocks m_s halves after every step, after a finite number of steps S, we obtain a single square block. Its final QR-decomposition gives the lexicographically first column base marking the desired columns of $\boldsymbol{A}$ to obtain B_{t+1}. During this process we perform at most $\sum_{s=0}^{S} m_s = \sum_{s=0}^{S} \lceil m_0 \cdot 2^{-s} \rceil \leqslant 3m_0$ decompositions. Each decomposition has cost $\mathcal{O}((2|V_a|)^{\omega+\epsilon})$, as shown by Demmel et al. [2007], for any $\epsilon > 0$, where $\omega < 3$ is the exponent of matrix multiplication.

Finally, accounting for all row partitions and considering for a suitable constant C, we have

$$\sum_{a \in \Sigma} 3\lceil n/|V^{(0)}_a| \rceil (2|V^{(0)}_a|)^{\omega+\epsilon} \leqslant n \sum_{a \in \Sigma} C|V_a|^{\omega+\epsilon-1} \leqslant n \cdot Cn^{\omega+\epsilon-1} \leqslant Cn^{\omega+\epsilon}.$$

Therefore, to obtain B_{t+1} from B_t, we perform $\mathcal{O}(n^{\omega+\epsilon})$ operations and $\mathcal{O}(Tn^{\omega+\epsilon})$ in total.

□

Note that the last bound in the above proof of Lemma A.8 is both of theoretical and practical interest. The QR-decomposition is a standard building block of many numerical libraries (e.g., LAPACK and SciPy), providing highly optimized implementations available. Moreover, splitting wide submatrix into a batch of square blocks allows for highly parallel processing of the algorithm on modern hardware architectures.

Theorem 3.1. *Let χ be a Σ-coloring on a graph G with n vertices, and let T in $\mathbb{N}$ be a limit. Then the canonical subsets $(S_t)_{t=0}^{T}$, (satisfying (i), (ii), and (iii)), are computable in time $\mathcal{O}(Tn^{\omega+\epsilon})$ for any $\epsilon > 0$ where ω is the exponent of matrix multiplication.*

Proof. We use the Algorithm 1, which is correct by Lemma A.7 and satisfying the time complexity by Lemma A.8. □

In the following part, we prove two statements from the end of Section 3. For $t \geqslant 0$, we recall that the columns of matrix $\boldsymbol{B}_t = \boldsymbol{B}_t(G, \chi)$ span the same space as the columns of the walk incidence matrix $\boldsymbol{W}_t(G, \chi)$. The columns of $\boldsymbol{B}_t$ are indexed by the words in $S_t(G, \chi)$. Let us denote by $\boldsymbol{z}_t = 00 \cdots 0$ a constant word of length t.

Observation 3.2. *Let χ_{triv} be the $\{0\}$-coloring on a graph G with at least one edge. Then for every $t \geqslant 0$: (a) it holds that $|S_t| = 1$; (b) the only entries, $\boldsymbol{C}^{\boldsymbol{I}}_t[\boldsymbol{z}_t, \boldsymbol{z}_{t+1}] = \frac{\mathbf{1}^\top \boldsymbol{A}^{2t+1}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1}}$, and $\boldsymbol{C}^{\boldsymbol{A}}_t[\boldsymbol{z}_t, \boldsymbol{z}_{t+1}] = \frac{\mathbf{1}^\top \boldsymbol{A}^{2t+2}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1}}$.*

Proof. The assumption that G has at least one (undirected) edge ensures that the denominators are well defined. Indeed, G then contains at least one walk of every length t, that is, $\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1} \geqslant 1$.

The first part (a) follows from the fact that the only word in $S_t(G, \chi_{\mathrm{triv}})$ is $\boldsymbol{z}_t$, and thus every queue Q_{t+1} of the Algorithm 1 contains exactly the one word $\boldsymbol{z}_{t+1}$. Note that the only partition matrix $\boldsymbol{P}_a = I$.

For the second part (b), we follow the definition of the matrices $\boldsymbol{C}^{\boldsymbol{I}}_t$ and $\boldsymbol{C}^{\boldsymbol{A}}_t$ in $\mathbb{R}^{\{\boldsymbol{z}_t\} \times \{\boldsymbol{z}_{t+1}\}}$:

$$\begin{aligned}
\boldsymbol{C}^{\boldsymbol{M}}_t[\boldsymbol{z}_t, \boldsymbol{z}_{t+1}] &= \mathbf{1}\boldsymbol{C}^{\boldsymbol{M}}_t\mathbf{1}^\top = \boldsymbol{B}^{+}_t \boldsymbol{M} \boldsymbol{B}_{t+1} = \\
&= ((\boldsymbol{P}_a \boldsymbol{A} \boldsymbol{P}_a)^t \mathbf{1})^{+} \boldsymbol{M} ((\boldsymbol{P}_a \boldsymbol{A} \boldsymbol{P}_a)^{t+1} \mathbf{1}) = \\
&= (\boldsymbol{A}^t \mathbf{1})^{+} \boldsymbol{M} (\boldsymbol{A}^{t+1} \mathbf{1}) = \\
&= ((\boldsymbol{A}^t \mathbf{1})^\top \boldsymbol{A}^t \mathbf{1})^{-1} \cdot (\boldsymbol{A}^t \mathbf{1})^\top \boldsymbol{M} \boldsymbol{A}^t \mathbf{1} = \\
&= (\mathbf{1}^\top \boldsymbol{A}^{2t} \mathbf{1})^{-1} \cdot \mathbf{1}^\top \boldsymbol{A}^t \boldsymbol{M} \boldsymbol{A}^{t+1} \mathbf{1}.
\end{aligned}$$

By setting $\boldsymbol{M} := \boldsymbol{I}$ we obtain the result for the $\boldsymbol{C}^{\boldsymbol{I}}_t$, and by $\boldsymbol{M} := \boldsymbol{A}$ for the $\boldsymbol{C}^{\boldsymbol{A}}_t$. □

Proposition 3.3. *Let χ_{id} be the V-coloring on a graph $G = (V, E)$ with n vertices. By $\boldsymbol{v}_{t,u}$, we denote the (unique) word in S_t with the last color u in V. Then for every $t \geqslant 1$: (a) it holds that $|S_0| = 1$, and $|S_t| = |V|$; (b) for entries $\boldsymbol{C}_0^{\boldsymbol{I}}[\lambda, u] = \frac{1}{n}$, and $\boldsymbol{C}_t^{\boldsymbol{I}}[\boldsymbol{v}_{t,u}, \boldsymbol{v}_{t+1,v}] = \boldsymbol{I}[u, v]$ and $\boldsymbol{C}_t^{\boldsymbol{A}}[\boldsymbol{v}_{t,u}, \boldsymbol{v}_{t+1,v}] = \boldsymbol{A}[u, v]$.*

Proof. We denote the vertices V by $\{u_1, u_2, \ldots, u_n\}$, which here coincides with the set of colors. To prove (a), we proceed by induction on t. For the base case $t = 0$, we have trivially $|S_0| = 1$, as $S_0 = \{\lambda\}$. For the case $t = 1$, we have $S_1 = V$, since $\boldsymbol{P}_{u_i} = e_{u_i} e_{u_i}^\top$ for u_i in V, and furthermore, u_i-th column of $\boldsymbol{B}_1$ is $\boldsymbol{P}_{u_i}\mathbf{1} = e_{u_i}$, and thus we have $\boldsymbol{B}_1 = \boldsymbol{I}$. For the induction step, we assume that $S_t = \{\boldsymbol{v}_{t,u} \mid u \text{ in } V\}$. Since $\boldsymbol{W}_{t+1}[-, \boldsymbol{w}u] = \boldsymbol{P}_u \boldsymbol{A} \boldsymbol{W}_t[-, \boldsymbol{w}]$, there is for each u in V a unique word $\boldsymbol{w}u$ in S_{t+1}, and the base $\mathcal{B}_{t+1}$ is also canonical, that is, $\boldsymbol{B}_{t+1} = \boldsymbol{I}$. Thus, we have $|S_{t+1}| = |V|$.

For the second part (b), we have $\boldsymbol{C}_0^{\boldsymbol{I}} = \mathbf{1}^+ \boldsymbol{I} \boldsymbol{B}_1 = (\mathbf{1}^\top \mathbf{1})^{-1} \mathbf{1}^\top \boldsymbol{I} \boldsymbol{I} = \frac{1}{n} \mathbf{1}^\top$. Next, for $t \geqslant 1$ and $\boldsymbol{M}$ in $\{\boldsymbol{A}, \boldsymbol{I}\}$, we get $\boldsymbol{C}_t^{\boldsymbol{M}} = \boldsymbol{B}_t^+ \boldsymbol{M} \boldsymbol{B}_{t+1} = \boldsymbol{I}^+ \boldsymbol{M} \boldsymbol{I} = \boldsymbol{M}$. □

A.3 Constraints on Walk-Incidence Matrices

In the main text, we have noted that the matrices $\boldsymbol{W}_t(G, \chi)$ are not completely arbitrary. We now show in detail, how the structure of the matrices $\boldsymbol{W}_t(G, \chi)$ is influenced by the weighted finite automata $\mathcal{A}(G, \chi)$. Suppose we have two graphs G and H, and that the automata $\mathcal{A}(G, \chi)$ and $\mathcal{A}(H, \chi)$ are equivalent, $[\![\mathcal{A}(G, \chi)]\!] = [\![\mathcal{A}(H, \chi)]\!]$. Then there is a minimal weighted automaton $\mathcal{A}_1$ common for $\mathcal{A}(G, \chi)$ and $\mathcal{A}(H, \chi)$. By Equation 11 and Equation 12, there are matrices $\boldsymbol{F}^G$ in $\mathbb{R}^{V(G) \times Q_1}$ and $\boldsymbol{F}^H$ in $\mathbb{R}^{V(H) \times Q_1}$ mapping automata induced by G and H to the common minimal automata $\mathcal{A}_1$. Here, we denote the matrix between the two automata $\mathcal{A}(G, \chi)$ and $\mathcal{A}(H, \chi)$ by the following:

$$\boldsymbol{F} = (\boldsymbol{F}^G)(\boldsymbol{F}^H)^\top \text{ in } \mathbb{R}^{V(G) \times V(H)}.$$

Note that $\boldsymbol{F}\mathbf{1} = \boldsymbol{F}^G (\boldsymbol{F}^H)^\top \mathbf{1} = \boldsymbol{F}^G \mathbf{1} = \mathbf{1}$, and similarly $\mathbf{1}^\top \boldsymbol{F} = \mathbf{1}^\top$.

Lemma A.9. *Using the notation above, we have for every $t \geqslant 0$:*

$$\boldsymbol{W}_t(G, \chi) = \boldsymbol{F} \boldsymbol{W}_t(H, \chi).$$

Proof. As the Σ-coloring χ is fixed, we write $\boldsymbol{W}_t^G = \boldsymbol{W}_t(G, \chi)$ and $\boldsymbol{W}_t^H = \boldsymbol{W}_t(H, \chi)$. In addition, we denote the matrices of the first graph $\boldsymbol{A}^G$, $\boldsymbol{P}_a^G$, and similarly for the second graph $\boldsymbol{A}^H$, $\boldsymbol{P}_a^H$, given that a in Σ. We proceed by induction on t, proving the following statement for each word $\boldsymbol{w}$ in Σ^t: $\boldsymbol{W}_t^G[-, \boldsymbol{w}] = \boldsymbol{F} \boldsymbol{W}_t^H[-, \boldsymbol{w}]$.

For $t = 0$, from Lemma A.5 it follows that

$$\boldsymbol{W}_0^G[-, \lambda] = \mathbf{1} = \boldsymbol{F}\mathbf{1} = \boldsymbol{F} \boldsymbol{W}_0^H[-, \lambda].$$

Moreover, using Equation 11, for $t = 1$, we get for any a in Σ

$$\boldsymbol{W}_1^G[-, a] = \boldsymbol{P}_a^G \mathbf{1} = \boldsymbol{P}_a^G \boldsymbol{F}\mathbf{1} = \boldsymbol{F} \boldsymbol{P}_a^H \mathbf{1} = \boldsymbol{F} \boldsymbol{W}_1^H[-, a].$$

For the induction step, we assume that the lemma holds for all words of length t. We take a word $\boldsymbol{w}a$ of length $t + 1$, which denotes the last color of $\boldsymbol{w}$ by b. It follows:

$$\begin{aligned}
\boldsymbol{W}_{t+1}^G[-, \boldsymbol{w}a] &= \boldsymbol{P}_a^G \boldsymbol{A}^G \boldsymbol{P}_b^G \boldsymbol{W}_t^G[-, \boldsymbol{w}] = \\
&= \boldsymbol{P}_a^G \boldsymbol{A}^G \boldsymbol{P}_b^G F \boldsymbol{W}_t^H[-, \boldsymbol{w}] = \\
&= \boldsymbol{F} \boldsymbol{P}_a^H \boldsymbol{A}^H \boldsymbol{P}_b^H \boldsymbol{W}_t^H[-, \boldsymbol{w}] = \\
&= \boldsymbol{F} \boldsymbol{W}_{t+1}^H[-, \boldsymbol{w}a],
\end{aligned}$$

which finishes the proof. □

Characterization of Reduced Matrices. We recall that for graph G and a coloring χ, the *graph invariant* of reduced matrices $\mathcal{I}_{\mathrm{R}}(G, \chi)$ is defined as the set of pairs of matrices

$$\mathcal{I}_{\mathrm{R}}(G, \chi) = \left\{ \left(\boldsymbol{C}_t^{\boldsymbol{A}}, \boldsymbol{C}_t^{\boldsymbol{I}} \right) \mid 0 \leqslant t < n \right\},$$

to state the following lemma:

Lemma A.10. *For any coloring χ, we have $\mathcal{I}_R(-,\chi) \sqsubseteq [\![\mathcal{A}(-,\chi)]\!]$.*

Proof. Suppose we have two graphs G and H that are not distinguished by the semantics of their induced automata. Then there is a matrix $\boldsymbol{F}$ in $\mathbb{R}^{V(G)\times V(H)}$ adjoining these two automata. Given a Σ-coloring χ, and two graphs G, H we have from the Lemma A.9 that $\boldsymbol{B}_t^G = \boldsymbol{F}\boldsymbol{B}_t^H$ for all $t \geqslant 0$. For $\boldsymbol{M}$ in $\{\boldsymbol{A}, \boldsymbol{I}\}$, we have

$$\begin{aligned}
\boldsymbol{C}_t^{\boldsymbol{M}}(G,\chi) &= (\boldsymbol{B}_t^G)^+ \boldsymbol{M}^G \boldsymbol{B}_{t+1}^G = (\boldsymbol{B}_t^G)^+ \boldsymbol{M}^G \boldsymbol{F} \boldsymbol{B}_{t+1}^H \\
&= (\boldsymbol{B}_t^G)^+ \boldsymbol{F}\boldsymbol{M}^H \boldsymbol{B}_{t+1}^H = (\boldsymbol{B}_t^G)^+ \boldsymbol{F} \left(\boldsymbol{B}_t^H (\boldsymbol{B}_t^H)^+\right) \boldsymbol{M}^H \boldsymbol{B}_{t+1}^H \\
&= (\boldsymbol{B}_t^G)^+ \boldsymbol{F}\boldsymbol{B}_t^H (\boldsymbol{B}_t^H)^+ \boldsymbol{M}^H \boldsymbol{B}_{t+1}^H = (\boldsymbol{B}_t^G)^+ \left(\boldsymbol{F}\boldsymbol{B}_t^H\right) (\boldsymbol{B}_t^H)^+ \boldsymbol{M}^H \boldsymbol{B}_{t+1}^H \\
&= (\boldsymbol{B}_t^G)^+ \left(\boldsymbol{B}_t^G\right) (\boldsymbol{B}_t^H)^+ \boldsymbol{M}^H \boldsymbol{B}_{t+1}^H = (\boldsymbol{B}_t^H)^+ \boldsymbol{M}^H \boldsymbol{B}_{t+1}^H.
\end{aligned}$$

The final expression is equal to $\boldsymbol{C}_t^{\boldsymbol{M}}(H,\chi)$, which finishes the proof. □

In the previous lemma, we have shown that the reduced matrices are invariant under the equivalence of the automata. To state the other direction, we first give a simpler lemma for $\chi = \chi_{\text{triv}}$ to illustrate the structure of the more general lemma.

Lemma A.11. *Let χ_{triv} be the trivial coloring, then $\mathcal{I}_R(-,\chi_{\text{triv}}) \sqsupseteq [\![\mathcal{A}(-,\chi_{\text{triv}})]\!]$.*

Proof. As shown in Observation 3.2, the matrices $\boldsymbol{C}_t^{\boldsymbol{I}}(G,\chi_{\text{triv}})$ and $\boldsymbol{C}_t^{\boldsymbol{A}}(G,\chi_{\text{triv}})$ are of a single entry $(\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1})^{-1}\mathbf{1}^\top \boldsymbol{A}^{2t+1}\mathbf{1}$ and $(\mathbf{1}^\top \boldsymbol{A}^{2t}\mathbf{1})^{-1}\mathbf{1}^\top \boldsymbol{A}^{2t+2}\mathbf{1}$, respectively. We let the symbol $a = 0$, so that χ_{triv} is the $\{a\}$-coloring on G. Note that $|V(G)| = n = |\mathcal{I}_{\mathrm{R}}|$.

Next, the transition matrices of $\mathcal{A}(G,\chi_{\text{triv}})$ are $\boldsymbol{M}(a) = \boldsymbol{P}_a = I$, and $\boldsymbol{M}(a|a) = \boldsymbol{P}_a\boldsymbol{A}\boldsymbol{P}_a = \boldsymbol{A}$, and generally for k-th repetition of a, $\boldsymbol{M}(a|\cdots|a) = \boldsymbol{A}^k$. Thus, the semantics of the automata $\mathcal{A}(G,\chi_{\text{triv}})$ is determined by the formal series given as $a^k \mapsto \mathbf{1}^\top \boldsymbol{A}^k \mathbf{1}$.

For simplicity, we define the following three functions for all m in $\mathbb{N}$:

$$f(m) = \mathbf{1}^\top A^m \mathbf{1}, \quad g_1(m) = \frac{\mathbf{1}^\top \boldsymbol{A}^{2m+1}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{A}^{2m}\mathbf{1}}, \quad g_2(m) = \frac{\mathbf{1}^\top \boldsymbol{A}^{2m+2}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{A}^{2m}\mathbf{1}}.$$

In the language of such notation, it is sufficient to show that all values of f are determined by values of function g_1, g_2 and single value n. Moreover, it is sufficient to show values of $f(k)$ for $k < n$, since for values $k \geqslant n$, we can apply Cayley-Hamilton theorem and express A^k using the powers of A up to $n-1$.

We shall proceed by the following induction on k. For the base case $k = 0$, we have $f(0) = n$. For the induction step, assume that the value $f(k')$ is determined for each $k' < k$, we distinguished two cases: (a) $k = 2l+1$ is odd, and (b) $k = 2l+2$ is even.

For the odd case (a), we have $f(k) = f(2l+1) = f(2l)g_1(l)$, and for the even case (b), we have $f(k) = f(2l+2) = f(2l)g_2(l)$. Since $f(2l)$ is known from the induction hypothesis, we can compute $f(k)$ from $f(2l)$ and $g_1(l)$ or $g_2(l)$. □

Here, before we state a more general variant of Lemma A.11 for any coloring χ, we introduce the following notation. Let χ be a $\overline{\Sigma}$-coloring on a graph G, and $\mathcal{I}_{\mathrm{R}}(G,\chi)$ the invariant of reduced matrices $\boldsymbol{C}_t^{\boldsymbol{A}}(G,\chi)$, $\boldsymbol{C}_t^{\boldsymbol{I}}(G,\chi)$ in $\mathbb{R}^{S_t \times S_{t+1}}$ for $0 \leqslant t < n$. Let a, b in Σ, and t in $\mathbb{N}$ such that $0 \leqslant t < n$, we define $\boldsymbol{M}_t(a)$ in $\mathbb{R}^{S_t \times S_{t+1}}$ by setting

$$\boldsymbol{M}_t(a)[\boldsymbol{w}_1 a, \boldsymbol{w}_2 a] \coloneqq \boldsymbol{C}_t^{\boldsymbol{I}}(G,\chi)[\boldsymbol{w}_1 a, \boldsymbol{w}_2 a], \tag{13}$$

for all $\boldsymbol{w}_1 a$ in S_t, and $\boldsymbol{w}_2 a$ in S_{t+1} and letting all other entries be zero. Similarly, we define $\boldsymbol{M}_t(a|b)$ in $\mathbb{R}^{S_t \times S_{t+1}}$ by setting

$$\boldsymbol{M}_t(a|b)[\boldsymbol{w}_1 a, \boldsymbol{w}_2 ab] \coloneqq \boldsymbol{C}_t^{\boldsymbol{A}}(G,\chi)[\boldsymbol{w}_1 a, \boldsymbol{w}_2 ab], \tag{14}$$

for all $\boldsymbol{w}_1 a, \boldsymbol{w}_2 ab$ in S_{t+1} and letting all other entries be zero.

Lemma A.12. *Let G be a graph and χ a Σ-coloring on G. Then for all a, b in Σ and each t in $\mathbb{N}$ it holds that*

$$\boldsymbol{M}_t(a) = \boldsymbol{B}_t^+ \boldsymbol{P}_a \boldsymbol{B}_{t+1}, \quad \text{and} \quad \boldsymbol{M}_t(a|b) = \boldsymbol{B}_t^+ \boldsymbol{P}_b \boldsymbol{A} \boldsymbol{P}_a \boldsymbol{B}_{t+1},$$

where $\boldsymbol{B}_t = \boldsymbol{B}_t(G, \chi)$.

Proof. We start with the proof of the first identity. Let us fix t in $\mathbb{N}$, then

$$\boldsymbol{C}_t^I(G, \chi) = \boldsymbol{B}_t^+ (\textstyle\sum_a \boldsymbol{P}_a) \boldsymbol{B}_{t+1} = \displaystyle\sum_{a \in \Sigma} \boldsymbol{B}_t^+ \boldsymbol{P}_a \boldsymbol{B}_{t+1} = \sum_{a \in \Sigma} \boldsymbol{M}_t(a). \tag{15}$$

On the other hand, since $\boldsymbol{B}_t[-, \boldsymbol{w}_1 a] = (\boldsymbol{P}_a \boldsymbol{B}_t)[-, \boldsymbol{w}_1 a]$ for each $\boldsymbol{w}_1 a$ in S_t, we get for each $\boldsymbol{w}_1 x$ in S_t, and each $\boldsymbol{w}_2 y$ in S_{t+1}, that it holds

$$((\boldsymbol{B}_t^\top \boldsymbol{B}_t)^{-1} \boldsymbol{B}_t^\top \boldsymbol{P}_a \boldsymbol{B}_{t+1})[\boldsymbol{w}_1 x, \boldsymbol{w}_2 y] = ((\boldsymbol{B}_t^\top \boldsymbol{B}_t)^{-1} \boldsymbol{B}_t^\top \boldsymbol{P}_x^\top \boldsymbol{P}_a \boldsymbol{P}_y \boldsymbol{B}_t)[\boldsymbol{w}_1 x, \boldsymbol{w}_2 y],$$

By Proposition A.2 it follows that the expression is zero if $x \neq a$ or $y \neq a$. In conjunction with Eq. (15), it follows that left-hand side and right-hand side coincide entry-wise and thus $\boldsymbol{M}_t(a) = \boldsymbol{B}_t^+ \boldsymbol{P}_a \boldsymbol{B}_{t+1}$. The latter identity, we show similarly by fixing t in $\mathbb{N}$,

$$\begin{aligned} \boldsymbol{C}_t^A(G, \chi) &= \boldsymbol{B}_t^+ (\textstyle\sum_{a,b} \boldsymbol{P}_b \boldsymbol{A} \boldsymbol{P}_a) \boldsymbol{B}_{t+1} \\ &= \sum_{a,b \in \Sigma} \boldsymbol{B}_t^+ \boldsymbol{P}_x \boldsymbol{P}_b \boldsymbol{A} \boldsymbol{P}_a \boldsymbol{P}_y \boldsymbol{B}_{t+1} = \sum_{a,b \in \Sigma} \boldsymbol{M}_t(a|b). \end{aligned} \tag{16}$$

Similarly, we have for each $\boldsymbol{w}_1 x$ in S_t, and each $\boldsymbol{w}_2 y$ in S_{t+1},

$$(\boldsymbol{B}_t^+ \boldsymbol{P}_a \boldsymbol{A} \boldsymbol{P}_b \boldsymbol{B}_{t+1})[\boldsymbol{w}_1 x, \boldsymbol{w}_2 y] = (\boldsymbol{B}_t^+ \boldsymbol{P}_x^\top \boldsymbol{P}_a \boldsymbol{A} \boldsymbol{P}_b \boldsymbol{P}_y \boldsymbol{B}_t)[\boldsymbol{w}_1 x, \boldsymbol{w}_2 y],$$

which is zero if $x \neq a$ or $y \neq b$. In conjunction with Eq. (16), it follows that left-hand side and right-hand side coincide entry-wise and thus $\boldsymbol{M}_t(a|b) = \boldsymbol{B}_t^+ \boldsymbol{P}_b \boldsymbol{A} \boldsymbol{P}_a \boldsymbol{B}_{t+1}$. □

Lemma A.13. *For any coloring χ, we have $\mathcal{I}_R(-, \chi) \sqsupseteq [\![\mathcal{A}(-, \chi)]\!]$.*

Proof. We first show that reduced matrices $\mathcal{I}_{\mathrm{R}} = \mathcal{I}_{\mathrm{R}}(G, \chi)$, for some graph G, encode a specific layered computation of minimal automata with the semantics of $\mathcal{A}(G, \chi)$. A direct consequence of Lemma A.12 is that by selecting the entries of reduced matrices in $\mathcal{I}_{\mathrm{R}}$ as in Eq. (13) and Eq. (14), we can obtain $\boldsymbol{M}_t(a)$ and $\boldsymbol{M}_t(a|b)$ for all a, b in Σ and where $0 \leqslant t < n$.

From the size of $\mathcal{I}_{\mathrm{R}}$, we get $n = |\mathcal{I}_{\mathrm{R}}| = |V(G)|$. For a word $\boldsymbol{w} = w_1 w_2 \cdots w_{t+1}$ in Σ^{t+1}, we define the expression $\gamma_{\boldsymbol{w}}(\mathcal{I}_{\mathrm{R}})$ in $\mathbb{R}^{\{\lambda\} \times S_{t+1}}$ as follows:

$$\gamma_{\boldsymbol{w}}(\mathcal{I}_{\mathrm{R}}) = n \cdot \boldsymbol{M}_0(w_1|w_2) \boldsymbol{M}_1(w_2|w_3) \cdots \boldsymbol{M}_t(w_t|w_{t+1}).$$

Next, we take $\boldsymbol{B}_t = \boldsymbol{B}_t(G, \chi)$ and $\boldsymbol{M} \colon \Sigma \to \mathbb{R}^{V \times V}$ the transition matrices of $\mathcal{A}(G, \chi)$, it follows from Lemma A.12 that

$$\begin{aligned} \gamma_{\boldsymbol{w}}(\mathcal{I}_{\mathrm{R}}) &= n \boldsymbol{B}_0^+ \boldsymbol{P}_{w_1} \boldsymbol{A} \boldsymbol{P}_{w_2} \boldsymbol{B}_1 \boldsymbol{B}_1^+ \boldsymbol{P}_{w_2} \boldsymbol{A} \boldsymbol{P}_{w_3} \boldsymbol{B}_2 \boldsymbol{B}_2^+ \cdots \boldsymbol{P}_{w_t} \boldsymbol{A} \boldsymbol{P}_{w_{t+1}} \boldsymbol{B}_t \\ &= n \mathbf{1}^+ \boldsymbol{P}_{w_1} \boldsymbol{A} \boldsymbol{P}_{w_2} \cdots \boldsymbol{P}_{w_t} \boldsymbol{A} \boldsymbol{P}_{w_{t+1}} \boldsymbol{B}_t \\ &= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}) \boldsymbol{B}_t, \end{aligned}$$

where $\overline{\boldsymbol{w}} = w_1|w_2|\cdots|w_{t+1}$ in $\overline{\Sigma}^{t+1}$.

We prove for every t, such that $0 \leqslant t < n$, that weight of any word $\overline{\boldsymbol{w}}$ in $\overline{\Sigma}^t$, $[\![\mathcal{A}(G, \chi)]\!](\overline{\boldsymbol{w}}) = \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}) \mathbf{1}$ can be expressed as a linear combination of γ-vectors that depend only on $\mathcal{I}_{\mathrm{R}}(G, \chi)$. We shall use the following notation, for the word $\boldsymbol{w} = w_1 w_2 \cdots w_m$ in Σ^m, we denote the word $w_1|w_2|\cdots|w_m$ in $\overline{\Sigma}^m$ by $\overline{\boldsymbol{w}}$, and the reverse words $w_m w_{m_1} \cdots w_1$ in Σ^m by $\boldsymbol{w}^R$, and finally, $w_m|w_{m-1}|\cdots|w_1$ in $\overline{\Sigma}^m$ by $\overline{\boldsymbol{w}}^R$.

For the base case $t = 0$, we have $\overline{\boldsymbol{w}} = \lambda$, and thus $\mathbf{1}^T \boldsymbol{M}(\lambda) \mathbf{1} = \mathbf{1}^T \mathbf{1} = n$, which is equal to $\gamma_\lambda(\mathcal{I}_{\mathrm{R}}) \mathbf{1} = n \mathbf{1}^+ \mathbf{1} = n$.

In the induction step $t \geqslant 1$, we distinguish two cases: (a) $t = 2l$ is even, and (b) $t = 2l + 1$ is odd. For the even case (a), we write the given word as $\overline{\boldsymbol{w}} = \overline{\boldsymbol{w}}_p|a|\overline{\boldsymbol{w}}_s$, for the suitable $\boldsymbol{w}_p$ and $a\boldsymbol{w}_s$ in Σ^l.

Since the columns of $\boldsymbol{W}_l = \boldsymbol{W}_l(G, \chi)$ are spanned by the columns of $\boldsymbol{B}_l$, there is a vector $\boldsymbol{x}$ in $\mathbb{R}^{S_l}$ such that by Lemma A.5 it holds that

$$\boldsymbol{M}(a|\overline{\boldsymbol{w}}_s)\mathbf{1} = \boldsymbol{W}_l[-, \boldsymbol{w}_s^R a] = \boldsymbol{B}_l \boldsymbol{x}.$$

Note that the vector $\boldsymbol{x}$ is independent of the choice of the base of the transition matrices $\boldsymbol{M}$, as for any base-changing matrix $\boldsymbol{F}$ in $\mathbb{R}^{V(H)\times V(G)}$, for some graph H with $\mathcal{I}_{\mathrm{R}}(H, \chi) = \mathcal{I}_{\mathrm{R}}$, it holds $\boldsymbol{W}_l(H, \chi) = \boldsymbol{F}\boldsymbol{W}_l = \boldsymbol{F}\boldsymbol{B}_l\boldsymbol{x} = \boldsymbol{B}_l(H, \chi)\boldsymbol{x}$. Next, we obtain

$$\begin{aligned}
\mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}})\mathbf{1} &= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}_p|a \cdot a|\overline{\boldsymbol{w}}_s)\mathbf{1} \\
&= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}_p|a)\boldsymbol{M}(a|\overline{\boldsymbol{w}}_s)\mathbf{1} \\
&= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}_p|a)\boldsymbol{B}_l\boldsymbol{x} = \gamma_{\overline{\boldsymbol{w}}_p|a}(\mathcal{I}_{\mathrm{R}})\boldsymbol{x}.
\end{aligned}$$

Similarly, for the odd case (b), we write $\overline{\boldsymbol{w}} = \overline{\boldsymbol{w}}_p|a|\overline{\boldsymbol{w}}_s$, for the suitable $\boldsymbol{w}_p$ in Σ^l and $a\boldsymbol{w}_s$ in Σ^{l+1}. Analogously, we can find a vector $\boldsymbol{x}$ in $\mathbb{R}^{S_{l+1}}$ such that $\boldsymbol{M}(a|\overline{\boldsymbol{w}}_s)\mathbf{1} = \boldsymbol{B}_{l+1}\boldsymbol{x}$ to obtain

$$\begin{aligned}
\mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}})\mathbf{1} &= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}_p|a \cdot a \cdot a|\overline{\boldsymbol{w}}_s)\mathbf{1} \\
&= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}_p|a)\boldsymbol{M}(a)\boldsymbol{M}(a|\overline{\boldsymbol{w}}_s)\mathbf{1} \\
&= \mathbf{1}^T \boldsymbol{M}(\overline{\boldsymbol{w}}_p|a)\boldsymbol{M}(a)\boldsymbol{B}_{l+1}\boldsymbol{x} \\
&= \gamma_{\overline{\boldsymbol{w}}_p|a}(\mathcal{I}_{\mathrm{R}})\boldsymbol{B}_l\boldsymbol{B}_l^{+}\boldsymbol{P}_a\boldsymbol{B}_{l+1}\boldsymbol{x} = \gamma_{\overline{\boldsymbol{w}}_p|a}(\mathcal{I}_{\mathrm{R}})\boldsymbol{M}_l(a)\boldsymbol{x}.
\end{aligned}$$

Thus, $\mathcal{I}_{\mathrm{R}} = \mathcal{I}_{\mathrm{R}}(G, \chi)$ determines the semantics of automata $\mathcal{A}(G, \chi)$. Moreover, $\mathcal{I}_{\mathrm{R}}(H, \chi) = \mathcal{I}_{\mathrm{R}}(G, \chi)$ implies $S_t(H, \chi) = S_t(G, \chi)$ for all $0 \leqslant t < n$, and hence the semantics of $\mathcal{A}(G, \chi)$ agrees with that of $\mathcal{A}(H, \chi)$. □

Theorem 4.2. *For every coloring χ, it holds that $\mathcal{I}_R(-, \chi) \equiv \mathsf{wr}(-, \chi)$.*

Proof. The proof follows from Lemma A.13 and Lemma A.10. □

B Homomorphism Counts and Colored Walks

This section contains the proofs for the statements of Section 4.1. We first show that homomorphism counts from caterpillars $\mathcal{C}_1$ into a given graph can be characterized by walk incidences with respect to the coloring $\chi_{\deg}$. Our initial argument employs elementary techniques that are later generalized using methods from quantum graphs [Lovász and Szegedy, 2009, Dvořák, 2010] and logic (cf. [Immerman and Lander, 1990, Cai et al., 1992, Grohe, 2017]).



Figure 10: Section 4 establishes the equivalence between homomorphism counts over caterpillars and colored walk refinement with respect to (specific) colorings. First, we focus on folklore caterpillars (left), and then we generalize to (h, t)-caterpillars (right) in Section B.1. For every $h, t \geqslant 0$, we relate the following two functions on graphs: homomorphism counts over (h, t)-caterpillars $\mathsf{hom}(\mathcal{C}_{h,t}, -)$, and colored walk refinement $\mathsf{wr}^{(t)}(-, \chi_{\mathsf{cr}}^{(h)})$.

We begin by proving two technical lemmas.

Lemma B.1. *For every integer $t \geqslant 0$ and natural numbers $D_0, D_1, D_2, \ldots, D_t$ there are integral exponents $s_1, s_2 \ldots, s_t$ such that $D_0 \leqslant 2^{s_1}$, and such that all t-tuples of natural numbers $(d_1, d_2, \ldots, d_t)$ that satisfy $1 \leqslant d_i < D_i$ for i in $[t]$, are injectively represented by the product $p = d_1^{s_1} \cdot d_2^{s_2} \cdot \cdots \cdot d_t^{s_t}$.*

Proof. We can instead take logarithm of p, since $\log$ is an injective function,

$$\log_2 p = s_1 \log_2 d_1 + \cdots + s_t \log_2 d_t.$$

The values of $\log_2 d_i$ in $\{\log_2 1, \log_2 2, \ldots, \log_2(D_i - 1)\}$ are called *digits* for i in $[t]$. We construct a positional numeral system with a heterogeneous basis (i.e., $s_1, s_2, \ldots s_t$) special for each digit's position i. Since the intervals of digits at the i-th position are bounded by $\log_2 D_i$, we require to preserve the following inequality

$$s_{i+1} > \sum_{j=1}^{i} \log_2(D_j - 1) \cdot s_j.$$

To next meet the first condition $D_0 \leqslant 2^{s_1}$, we set $s_0 \coloneqq \lceil \log_2 D_0 \rceil$. Furthermore, we obtain $s_{i+1} \coloneqq 1 + \sum_{j=1}^{i} \lceil \log_2(D_j - 1) \rceil \cdot s_j$. Finally, we define $R_{t+1} = \log_2 p$ to inductively evaluate the following expression

$$\log_2 d_i = \frac{1}{s_i}\left(R_i - \sum_{j=0}^{i-1} s_j \log_2 d_j \right) \quad \text{where} \quad R_l = \frac{R_{l+1}}{s_{l+1}} - \left\lfloor \frac{R_{l+1}}{s_{l+1}} \right\rfloor,$$

for every l in $[t]$, and every i in $[t]$. Therefore, the values s_i are sufficient. □

Lemma B.2. *Let $\boldsymbol{x}$ be an m-tuple in $\mathbb{N}_{\geqslant 1}$ with mutually distinct elements, that is $\boldsymbol{x}[i] \neq \boldsymbol{x}[j]$ whenever $i \neq j$, and let m-tuples $\boldsymbol{a}$, $\boldsymbol{b}$ in $\mathbb{N}^m$ be such that $\boldsymbol{a} \neq \boldsymbol{b}$, that is $\boldsymbol{a}[i] \neq \boldsymbol{b}[i]$ for existing i in $[m]$. Then there is k in $\mathbb{N}$ such that*

$$\sum_{i=1}^{m} \boldsymbol{a}[i](\boldsymbol{x}[i])^k \neq \sum_{i=1}^{m} \boldsymbol{b}[i](\boldsymbol{x}[i])^k. \tag{17}$$

Proof. Denote tuple $\boldsymbol{a}$ by $(a_1, a_2, \ldots, a_m)$, $\boldsymbol{b}$ by $(b_1, b_2, \ldots, b_m)$, and the tuple $\boldsymbol{x}$ by $(x_1, x_2, \ldots, x_m)$. For a contradiction, suppose that for all k in $\mathbb{N}$ the equality in Equation 17 holds. Let us choose the index j such that $|a_j - b_j| > 0$ and, importantly, such that x_j is maximal (such j is unique since elements of $\boldsymbol{x}$ are distinct). Let us define functions f and g as follows

$$f(k) = \left|(a_j - b_j)x_j^k\right|, \quad g(k) = \Big|\sum_{i \neq j}(b_i - a_i)x_i^k\Big|. \tag{18}$$

As $k \to \infty$, the function f grows faster than the function g. Therefore, there exists k such that $f(k) > g(k)$, that is $(a_j - b_j)x_j^k > \sum_{i \neq j}(b_i - a_i)x_i^k$, which gives us the contradiction. □

In Proposition B.3, we prove the formula for counting homomorphisms from the class $\mathcal{C}_1$ that uses the leg sequence (see Section 4 and Figure 6) of the caterpillar graph. Next, a *star graph* is a rooted tree of height at most 1.

Given a $(1, t)$-caterpillar F in $\mathcal{C}_{1,t}$, we associate its leg sequence $(S_1, S_2, \ldots, S_t)$ of star graphs S_i, with the tuple $\boldsymbol{s}_F = (|S_1| - 1, |S_2| - 1, \ldots, |S_t| - 1)$ in $\mathbb{N}^t$, so that $|S_i| - 1$ is the number of leaves of the i-th leg of F. In terms of folklore caterpillars, the i-th entry of $\boldsymbol{s}_F$ is exactly the number of one-edge legs attached to the i-th vertex of the spine path. Importantly, every $(1, t)$-caterpillar graph F is fully described and determined by $\boldsymbol{s}_F$. Furthermore, we recall that $\mathsf{wr}^{(t)}(G, \chi_{\deg})$ is the multiset of all colored walks of length t in G.

Proposition B.3. *Let t in $\mathbb{N}$, then for every graph G it holds*

$$\mathsf{hom}(\mathcal{C}_{1,t}, G)[F] = \sum_{\boldsymbol{w} \text{ in } \mathbb{N}^t} \mathsf{wr}^{(t)}(\boldsymbol{w}) \cdot \prod_{i=1}^{t} \boldsymbol{w}[i]^{\boldsymbol{s}_F[i]}, \tag{19}$$

for each F in $\mathcal{C}_{1,t}$ with $\boldsymbol{s}_F$ in $\mathbb{N}^t$, where $\mathsf{wr}^{(t)}(\boldsymbol{w}) = \mathsf{wr}^{(t)}(G, \chi_{\deg})(\boldsymbol{w})$ for $\boldsymbol{w}$ in $\mathbb{N}^t$.

Proof. Given a colored walk $\boldsymbol{w}$ in $\mathbb{N}^t$, consider its occurrence $\boldsymbol{u} = (u_1, u_2, \ldots, u_t)$ in G. We denote the vertices of the spine of F that correspond to $\boldsymbol{s}_F$ by $l_1, l_2, \ldots, l_t$. Let $k_{\boldsymbol{u}}$ be the number of

graph homomorphisms $\varphi\colon F \to G$, mapping the spine of F to the occurrence $\boldsymbol{u}$, that is, such that $\varphi(l_i) = u_i$ for each i in $[t]$.

Therefore, any two distinct homomorphisms contributing to $k_{\boldsymbol{u}}$ differ precisely by their mapping of vertices outside the spine, namely, the leaves of the star graphs S_i for i in $[t]$.

Specifically, each leaf of S_i can be mapped to any vertex in its neighborhood $N(u_i)$, hence, we have exactly $\boldsymbol{w}[i] = \deg_G(u_i)$ choices. Because S_i contains $\boldsymbol{s}_F[i]$ such leaves, there are $\boldsymbol{w}[i]^{\boldsymbol{s}_F[i]}$ choices to map the legs of $\boldsymbol{P}_i$ independently of the other $\boldsymbol{P}_j$ ($j \neq i$). Therefore, we have $k_{\boldsymbol{u}} = \prod_{i=1}^{t} \boldsymbol{w}[i]^{\boldsymbol{s}_F[i]}$.

Finally, we sum over all independent $k_{\boldsymbol{u}}$ as follows:

$$\sum_{\boldsymbol{u} \text{ in } V(G)^t} k_{\boldsymbol{u}} = \sum_{\substack{\boldsymbol{u} \text{ occurrence} \\ \text{of } \boldsymbol{w} \text{ in } G}} \prod_{i=1}^{t} \boldsymbol{w}[i]^{\boldsymbol{s}_F[i]} = \sum_{\boldsymbol{w} \text{ in } \mathbb{N}^t} \mathsf{wr}^{(t)}(\boldsymbol{w}) \cdot \prod_{i=1}^{t} \boldsymbol{w}[i]^{\boldsymbol{s}_F[i]},$$

where in the second step, we distinguish occurrences $\boldsymbol{u}$ by their color $\boldsymbol{w}$, next, these occurrences are quantified by $\mathsf{wr}^{(t)}(\boldsymbol{w})$. Thus, we obtained an expression equal to the left-hand side of Equation 19. □

Theorem B.4. *For every t in $\mathbb{N}$, it holds that* $\mathsf{wr}^{(t)}(-, \chi_{\deg}) \equiv \mathsf{hom}(\mathcal{C}_{1,t}, -)$.

Proof. We first show $\mathsf{wr}^{(t)}(-, \chi_{\deg}) \sqsupseteq \mathsf{hom}(\mathcal{C}_{1,t}, -)$. Suppose that $\mathsf{wr}^{(t)}(G, \chi_{\deg}) = \mathsf{wr}^{(t)}(H, \chi_{\deg})$ for two given graphs G and H. Then by Equation 19 of Proposition B.3 we have that $\mathsf{hom}(\mathcal{C}_{1,t}, G)[F] = \mathsf{hom}(\mathcal{C}_{1,t}, H)[F]$ for every entry indexed by F in $\mathcal{C}_{1,t}$.

For the other direction, suppose that $\mathsf{wr}^{(t)}(G, \chi_{\deg}) \neq \mathsf{wr}^{(t)}(H, \chi_{\deg})$ for two given G and H. For clarity, we use the shortcut $m^X_{\boldsymbol{w}} = \mathsf{wr}^{(t)}(X, \chi_{\deg})(\boldsymbol{w})$ for graph X in $\{G, H\}$. Using this notation, there exists $\boldsymbol{w}'$ in $\mathbb{N}^t$ such that $m^G_{\boldsymbol{w}'} \neq m^H_{\boldsymbol{w}'}$.

Let n bound the number of vertices of G and H, then the maximum degree of both graphs is at most $n-1$. That means there are at most $(n-1)^t$ plain walks of length t in each graph, implying both

$$0 \leqslant m^G_{\boldsymbol{w}} \leqslant (n-1)^t, \quad \text{and} \quad 0 \leqslant m^H_{\boldsymbol{w}} \leqslant (n-1)^t.$$

Here, we use Lemma B.1, applied on t-tuples $\boldsymbol{w}$ in $[n-1]^t$. We choose $D_0 = n^t$ and bound entries by $D_1 = D_2 = \cdots = D_t = n$, to get injective representations of tuples in $[n]^t \times [n] \times \cdots \times [n]$. As a result, we obtain coefficients $(s_1, s_2, \ldots, s_t)$ such that the function

$$\big(m^X_{\boldsymbol{w}}, \boldsymbol{w}[1], \boldsymbol{w}[2], \ldots, \boldsymbol{w}[t]\big) \mapsto m^X_{\boldsymbol{w}} \cdot \boldsymbol{w}[1]^{s_1} \cdot \boldsymbol{w}[2]^{s_2} \cdots \boldsymbol{w}[t]^{s_t},$$

is injective provided that $m^X_{\boldsymbol{w}} > 0$ on both X in $\{G, H\}$.

In case either $m^G_{\boldsymbol{w}} = 0$ or $m^H_{\boldsymbol{w}} = 0$, choose $k = 1$ as a multiplier of coefficients $(s_1, \ldots, s_t)$ to satisfy Eq. (20) directly. Otherwise, to find suitable k, we apply Lemma B.2 by setting vectors $\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}$ in $[n-1]^t$ as follows:

$$\boldsymbol{x}[i] = \boldsymbol{w}_i[1]^{s_1} \cdot \boldsymbol{w}_i[2]^{s_2} \cdots \boldsymbol{w}_i[t]^{s_t}, \qquad \boldsymbol{a}[i] = m^G_{\boldsymbol{w}_i}, \qquad \boldsymbol{b}[i] = m^H_{\boldsymbol{w}_i},$$

where i is the index enumerating each $\boldsymbol{w}_i$ in $[n-1]^t$.

Since we assumed $m^G_{\boldsymbol{w}'} \neq m^H_{\boldsymbol{w}'}$, it holds $\boldsymbol{a} \neq \boldsymbol{b}$ and therefore by Lemma B.2 we can find a finite k such that

$$\sum_{\boldsymbol{w} \in [n-1]^t} m^G_{\boldsymbol{w}} \cdot \left(\prod_{i=1}^{t} \boldsymbol{w}[i]^{s_i} \right)^k \neq \sum_{\boldsymbol{w} \in [n-1]^t} m^H_{\boldsymbol{w}} \cdot \left(\prod_{i=1}^{t} \boldsymbol{w}[i]^{s_i} \right)^k. \tag{20}$$

Finally, there is a caterpillar F' in $\mathcal{C}_{1,t}$ determined by $\boldsymbol{s}_{F'} = (s_1 k, s_2 k, \ldots, s_t k)$, and since both sides of Eq. (20) can be rewritten by applying exponent k as $\mathsf{hom}(\mathcal{C}_{1,t}, G)[F'] \neq \mathsf{hom}(\mathcal{C}_{1,t}, H)[F']$ by Proposition B.3, we obtain $\mathsf{hom}(\mathcal{C}_{1,t}, G) \neq \mathsf{hom}(\mathcal{C}_{1,t}, H)$. □

B.1 Quantum Graph Homomorphisms

In this subsection, we extend the proof of Theorem B.4 to the case of generalized caterpillars $\mathcal{C}_{h,t}$. Similarly to the previous case in Proposition B.3, where we counted possible mappings of each leg of the folklore caterpillar, we can adapt a more general approach by replacing star graphs with 1-labeled graphs. Finally, we replace the counts of all possible mappings of star graphs by the homomorphism counts of linear combinations of 1-labeled graphs, called quantum 1-labeled graphs.

Labeled graphs. We follow the algebraic approach to quantum graphs by Lovász and Szegedy [2009], instantiating for the 1-labeled case. A 1-*labeled graph*, or simply *labeled graph*, $G^\bullet$ is a graph G with one distinguished vertex u in $V(G)$, called a *label*, denoted by $\mathsf{lab}(G^\bullet)$.

Let $\mathcal{F}^\bullet$ be a class of labeled graphs, $F^\bullet$ in $\mathcal{F}^\bullet$ labeled, and $G^\bullet$ another labeled graph then we define vector $\mathsf{hom}(\mathcal{F}^\bullet, G^\bullet)$ in $\mathbb{N}^{\mathcal{F}^\bullet}$ entry-wise: each entry $\mathsf{hom}(\mathcal{F}^\bullet, G^\bullet)[F^\bullet]$ is the number of graph homomorphisms $\varphi\colon V(G) \to V(H)$ that, moreover, preserve the label, i.e.,

$$\varphi(\mathsf{lab}(F^\bullet)) = \mathsf{lab}(G^\bullet).$$

In cases where we need to explicitly indicate the labeled vertex of $G^\bullet$, we write G^u for $u = \mathsf{lab}(G^\bullet)$ in $V(G)$.

Next, a *product* $G_1^\bullet \cdot G_2^\bullet$ of two labeled graphs $G_1^\bullet$, $G_2^\bullet$ is the graph created by identification of $\mathsf{lab}(G_1^\bullet)$ and $\mathsf{lab}(G_2^\bullet)$ in the disjoint union of $G_1^\bullet$ and $G_2^\bullet$.

A *quantum graph* is a formal linear combination of finitely many graphs. A 1-*labeled quantum graph*, or simply *labeled quantum graph*, is a formal linear combination (with real coefficients) of finitely many 1-labeled graphs. The homomorphism counting extends linearly to quantum graphs:

$$\mathsf{hom}(\mathcal{F}^\bullet,\, G^\bullet)\Big[\sum_{i=1}^{d} \alpha_i F_i^\bullet\Big] := \sum_{i=1}^{d} \alpha_i\, \mathsf{hom}(\mathcal{F}^\bullet, G^\bullet)[F_i^\bullet],$$

for the coefficients α_i in $\mathbb{R}$, and the 1-labeled graphs $F_i^\bullet$ for i in $[d]$.

Moreover, quantum graphs $G_1^\bullet, G_2^\bullet$ can be naturally combined by sum, product, and exponentiation operations. A *sum* $G_1^\bullet + G_2^\bullet$ is the sum of their linear combinations. A *product* $G_1^\bullet \cdot G_2^\bullet$ is the product of their linear combinations, where we use the definition of the product of two labeled graphs. Finally, an *exponentiation* $(G_1^\bullet)^k$ for an integer $k \geqslant 1$ is the k-fold product of $G^\bullet$ with itself. We will use a standard identity for 1-labeled, possibly quantum, graphs:

$$\mathsf{hom}(\mathcal{F}^\bullet, G^\bullet)[(F^\bullet)^k] = \left(\mathsf{hom}(\mathcal{F}^\bullet, G^\bullet)[F^\bullet]\right)^k. \tag{21}$$

The following result is due to Cai et al. [1992]. Let $\mathcal{C}_{2,h}$ denote the class of formulas of two-variable first-order logic with counting quantifiers, where the quantifier depth is bounded by h.

Theorem B.5 (Cai et al. [1992, Theorem 5.2])**.** *Let G^u, H^v be a pair of labeled graphs and $\chi_{\mathsf{cr}}^{(h)}$ be a coloring for h in $\mathbb{N}$. Then the following are equivalent:*

(i) $\chi_{\mathsf{cr}}^{(h)}(G, u) = \chi_{\mathsf{cr}}^{(h)}(H, v)$,

(ii) $(G, u) \equiv_{\mathcal{C}_{2,h}} (H, v)$.

The above result was followed by the work of Dvořák [2010] stating that the homomorphism counts of 1-labeled graphs are also equivalent to the first-order logic with counting quantifiers.

Theorem B.6 (Dvořák [2010, Theorem 7])**.** *Let G^u, H^v be a pair of labeled graphs, and let $\mathcal{T}_h^\bullet$ be the class of all 1-labeled trees of depth h. Then the following are equivalent*

(i) $(G, u) \equiv_{\mathcal{C}_{2,h}} (H, v)$,

(ii) $\mathsf{hom}(\mathcal{T}_h^\bullet, G) = \mathsf{hom}(\mathcal{T}_h^\bullet, H)$.

Proposition B.7. *Let h, t in $\mathbb{N}$, and let G be a graph and $\chi_{\mathsf{cr}}^{(h)}\colon V(G) \to \Sigma(G)$ a coloring. For each color c in $\Sigma(G)$, choose a vertex u in $V(G)$ such that $\chi_{\mathsf{cr}}^{(h)}(G, u) = c$, and denote by $G(c)$ the*

labeled graph G^u. Then, it holds:

$$\mathsf{hom}(\mathcal{C}_{h,t}, G)[F] = \sum_{\boldsymbol{w} \text{ in } \Sigma^t} \mathsf{wr}^{(t)}(G, \chi^{(h)}_{\mathsf{cr}})(\boldsymbol{w}) \cdot \prod_{i=1}^{t} \mathsf{hom}(\mathcal{T}^\bullet_h, G(\boldsymbol{w}[i]))[T^\bullet_i], \tag{22}$$

for every F in $\mathcal{C}_{h,t}$, where $(T^\bullet_1, T^\bullet_2, \dots T^\bullet_t)$ is a leg sequence of F corresponding to the spine $(\mathsf{lab}(T^\bullet_1), \mathsf{lab}(T^\bullet_2), \dots \mathsf{lab}(T^\bullet_t))$.

Proof. Given a colored walk $\boldsymbol{w}$ in $\mathbb{N}^t$, consider its occurrence $\boldsymbol{u} = (u_1, u_2, \dots, u_t)$ in G. Let $k_{\boldsymbol{u}}$ be the number of homomorphisms $\varphi \colon V(F) \to V(G)$ that map the spine of F to the occurrence $\boldsymbol{u}$, that is, $\mathsf{lab}(T_i) = u_i$ for i in $[t]$.

Specifically, for each u_i, independently, there are exactly $\mathsf{hom}(\mathcal{T}^\bullet_h, G^{u_i})[T^\bullet_i]$ ways to map the attached leg $T^\bullet_i$ into G:

$$\mathsf{hom}(\mathcal{C}_{h,t}, G)[F] = \sum_{\boldsymbol{u} \in V(G)^t} k_{\boldsymbol{u}} = \sum_{\substack{\boldsymbol{u} \text{ occurrence} \\ \text{of } \boldsymbol{w} \text{ in } G}} \prod_{i=1}^{t} \mathsf{hom}(\mathcal{T}^\bullet_h, G^{\boldsymbol{u}[i]})[T^\bullet_i]. \tag{23}$$

Furthermore, by the theorems Theorem B.5 and Theorem B.6, the number of tree homomorphisms $\mathsf{hom}(\mathcal{T}^\bullet_h, G^{u_i})[T^\bullet_i]$ only depends on the color of u_i, which is $\chi^{(h)}_{\mathsf{cr}}(G, u_i) = \boldsymbol{w}[i]$. Indeed we get, $\mathsf{hom}(\mathcal{T}^\bullet_h, G^{u_i})[T^\bullet_i] = \mathsf{hom}(\mathcal{T}^\bullet_h, G(\boldsymbol{w}[i]))[T^\bullet_i]$. Finally, reorganizing the sum in Eq. (23) to range over all possible colored walks $\boldsymbol{w}$ in Σ^t using the known multiplicities $\mathsf{wr}^{(t)}(G, \chi^{(h)}_{\mathsf{cr}})(\boldsymbol{w})$, we obtain exactly the expression on the right-hand side of Eq. (22). □

We also make use of the following result of Dvořák [2010], referred to as Lemma 6.

Proposition B.8 (Dvořák [2010, Lemma 6])**.** *For every formula $\psi(x)$ in $\mathcal{C}_{2,h}$, there exists a quantum graph $T^\bullet$ with its base in $\mathcal{T}^\bullet_h$ such that*

$$\mathsf{hom}(T^\bullet, G^u) = \begin{cases} 1, & \text{if } (G, u) \models \psi(x), \\ 0, & \text{otherwise.} \end{cases}$$

We are prepared to restate and prove the main result of this section.

Theorem 4.1. *For every $h, t \geqslant 0$, it holds that $\mathsf{wr}^{(t)}(-, \chi^{(h)}_{\mathsf{cr}}) \equiv \mathsf{hom}(\mathcal{C}_{h,t}, -)$.*

Proof. Following the structure of the proof of Theorem B.4, we first show that $\mathsf{wr}^{(t)}(-, \chi^{(h)}_{\mathsf{cr}}) \sqsupseteq \mathsf{hom}(\mathcal{C}_{h,t}, -)$. Suppose that $\mathsf{wr}^{(t)}(G, \chi^{(h)}_{\mathsf{cr}}) = \mathsf{wr}^{(t)}(H, \chi^{(h)}_{\mathsf{cr}})$ for two given graphs G and H. By Proposition B.7 we have that $\mathsf{hom}(\mathcal{C}_{h,t}, G)[F] = \mathsf{hom}(\mathcal{C}_{h,t}, H)[F]$ for every entry indexed by F in $\mathcal{C}_{h,t}$.

For the other direction, suppose that $\mathsf{wr}^{(t)}(G, \chi^{(h)}_{\mathsf{cr}}) \neq \mathsf{wr}^{(t)}(H, \chi^{(h)}_{\mathsf{cr}})$ for two given G and H. We want to prove the existence of a caterpillar in $\mathcal{C}_{h,t}$ for which the homomorphism counts differ. We denote the common finite set of colors given by $\chi^{(h)}_{\mathsf{cr}}$ in both graphs by $\Sigma = \Sigma(G) \cup \Sigma(H)$. Given a color c in Σ, we select a vertex u in $V(G) \sqcup V(H)$ such that $\chi^{(h)}_{\mathsf{cr}}(G', u) = c$, where G' is the disjoint union of G and H. We denote $G'^\bullet$ such that $\mathsf{lab}(G'^\bullet) = u$ by $G'(c)$.

Additionally, for each color c in Σ, there exists a formula ψ_c in $\mathcal{C}_{2,h}$ such that, for every graph G and vertex u in $V(G)$, we have $(G, u) \models \psi_c(x)$ if and only if $\chi^{(h)}_{\mathsf{cr}}(G, u) = c$. This follows from Claim 2 in the proof of Theorem 5.5.3 given in [Grohe, 2017]; see also [Cai et al., 1992, Immerman and Lander, 1990].

By Proposition B.8, it follows that for each $\psi_c(x)$ there is a quantum graph $T^\bullet_c$ of depth at most h such that $\mathsf{hom}(T^\bullet_c, G^u) = 1$ if $\psi_c(x)$ holds and 0 otherwise.

We fix a linear order on Σ by $\{c_1, c_2, \dots, c_p\} = \Sigma$, where $p = |\Sigma|$. For each color c_i there is a quantum graph $T^\bullet_{c_i}$ corresponding to ψ_{c_i}. We denote the quantum graph obtained by the scalar multiplication by i as $L^\bullet_{c_i} = (i+1) \cdot T^\bullet_{c_i}$, so that

$$\mathsf{hom}(\mathcal{T}^\bullet_h, G^u)[L^\bullet_{c_i}] = \begin{cases} i+1, & \text{if } \chi^{(h)}_{\mathsf{cr}}(G, u) = c_i, \\ 0, & \text{otherwise.} \end{cases}$$

Let n bound the number of vertices of G and H, let $m^X_{\boldsymbol{w}} = \mathsf{wr}^{(t)}(X, \chi^{(h)}_{\mathsf{cr}})(\boldsymbol{w})$ for X in $\{G, H\}$ and $\boldsymbol{w}$ in Σ^t. We apply Lemma B.1 with bounds $D_0 = n^t$, and $D_1 = \cdots = D_t = p + 2$ to get injective representations of tuples in $[n]^t \times [p+2] \times \cdots \times [p+2]$. As a result, we obtain coefficients $(s_1, s_2, \ldots, s_t)$ such that the function

$$\big(m^X_{\boldsymbol{w}}, \boldsymbol{w}[1], \boldsymbol{w}[2], \ldots, \boldsymbol{w}[t]\big) \mapsto m^X_{\boldsymbol{w}} \cdot \prod_{i=0}^{t} \big(\mathsf{hom}(\mathcal{T}^\bullet_h, G'(\boldsymbol{w}[i]))[L^\bullet_{\boldsymbol{w}[i]}]\big)^{s_i}$$

is injective provided that $m^X_{\boldsymbol{w}} > 0$ on both X in $\{G, H\}$.

Next, for each $\boldsymbol{w} = w_1 w_2 \ldots w_t$ in Σ^t, we construct a quantum caterpillar graph $F_{\boldsymbol{w}}$ with a sequence of t legs given by $\big((L^\bullet_{w_1})^{s_1}, (L^\bullet_{w_2})^{s_2}, \ldots, (L^\bullet_{w_t})^{s_t}\big)$.

In case either $m^G_{\boldsymbol{w}} = 0$ or $m^H_{\boldsymbol{w}} = 0$, choose $k = 1$ as a multiplier of coefficients $(s_1, \ldots, s_t)$ to satisfy Eq. (26) directly with $F^1_{\boldsymbol{w}}$ given by Eq. (24). Otherwise, to find suitable k, we apply Lemma B.2 by setting $\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}$ in $[p+2]^t$:

$$\boldsymbol{x}[i] = \prod_{j=0}^{t} (\mathsf{hom}(\mathcal{T}^\bullet_h, G'(\boldsymbol{w}_i[j]))[L^\bullet_{\boldsymbol{w}_i[j]}])^{s_j}, \quad \boldsymbol{a}[i] = m^G_{\boldsymbol{w}_i}, \quad \boldsymbol{b}[i] = m^H_{\boldsymbol{w}_i},$$

where i is the index enumerating each $\boldsymbol{w}_i$ in $[p+2]^t$. As a result we get sufficiently large exponent k distinguishing the expressions involving $\boldsymbol{a}$ and $\boldsymbol{b}$. Specifically, we consider the quantum caterpillar given by the spine

$$F^k_{\boldsymbol{w}} = \big((L^\bullet_{w_1})^{k \cdot s_1}, (L^\bullet_{w_2})^{k \cdot s_2}, \ldots, (L^\bullet_{w_t})^{k \cdot s_t}\big), \tag{24}$$

for which we obtain the following

$$\mathsf{hom}(\mathcal{C}_{h,t}, G)[F^k_{\boldsymbol{w}}] = (\mathsf{hom}(\mathcal{C}_{h,t}, G)[F_{\boldsymbol{w}}])^k \tag{25}$$

$$\neq (\mathsf{hom}(\mathcal{C}_{h,t}, H)[F_{\boldsymbol{w}}])^k = \mathsf{hom}(\mathcal{C}_{h,t}, H)[F^k_{\boldsymbol{w}}], \tag{26}$$

where both equalities follow from Proposition B.7 and the identity in Eq. (21).

Finally, because every quantum graph is a linear combination of non-quantum ones, it follows immediately that there exists at least one non-quantum caterpillar graph F' in $\mathcal{C}_{h,t}$ in the base of $F^k_{\boldsymbol{w}}$ for which

$$\mathsf{hom}(\mathcal{C}_{h,t}, G)[F'] \neq \mathsf{hom}(\mathcal{C}_{h,t}, H)[F'].$$

□

Corollary B.9. *For every $h \geqslant 0$ it holds that* $\mathsf{wr}(-, \chi^{(h)}_{\mathsf{cr}}) \equiv \mathsf{hom}(\mathcal{C}_h, -)$.

Proof. It follows from Theorem 4.1 and the fact that $\mathcal{C}_h$ is a superclass of $\mathcal{C}_{h,t}$ for t in $\mathbb{N}$. □

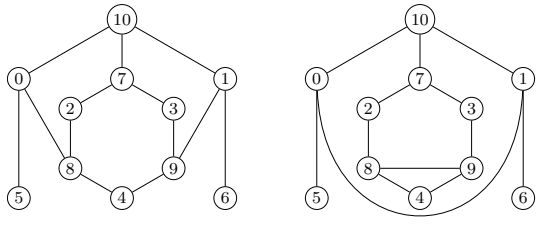

Figure 11: An example of a pair of graphs G and H that strictly separates the graph functions $\mathsf{hom}(\mathcal{C}_1, -)$ and $\mathsf{hom}(\mathcal{C}_0, -)$. Note that the graph H can be derived from G by removing the edges $\{0, 8\}$ and $\{1, 9\}$, and then adding the edges $\{0, 1\}$ and $\{8, 9\}$.

B.2 Strict separation

For a strict separation, we define closure under disjoint unions of generalized caterpillar graphs in order to apply existing theory on minor-closed classes. Formally, let $G = (V, E)$ be a graph and let $e = uv$ in E be an edge. The *contraction* of e in G is a graph G/e with removed e and identified vertices u and v. Graph G_m is a *minor* of G if it can be obtained by a sequence of edge deletions, vertex deletions and edge contractions from G.

A class of graphs $\mathcal{C}$ is *minor-closed* if for every G in $\mathcal{C}$ and every minor G' of G, we have G' in $\mathcal{C}$. We denote by $\mathcal{C}^{\sqcup}_{h,t}$ the closure of $\mathcal{C}_{h,t}$ under finite disjoint unions, that is, $\mathcal{C}^{\sqcup}_{h,t} \supseteq \mathcal{C}_{h,t}$ and if $(V_1, E_1), (V_2, E_2)$ in $\mathcal{C}^{\sqcup}_{h,t}$ then $(V_1 \sqcup V_2,\ E_1 \sqcup E_2)$ in $\mathcal{C}^{\sqcup}_{h,t}$. Analogously, we denote by $\mathcal{C}^{\sqcup}_{h}$ the closure of $\mathcal{C}_h$ under finite disjoint unions.

Proposition B.10. *For all integers $h, t \geqslant 0$, the class $\mathcal{C}^{\sqcup}_{h,t}$ is minor-closed.*

Proof. Take a graph G in $\mathcal{C}^{\sqcup}_{h,t}$ and consider either deletion or contraction of its edge uv. Edge uv lies in one of the connected components G', which lies in $\mathcal{C}_{h,t}$. Therefore, there is a sequence of legs $((L_1, s_1), \dots (L_t, s_t))$. We now consider following cases:

1. Contraction of edge in (L_i, s_i): we obtain by contraction $(L_i/uv, s_i)$ which is in $\mathcal{T}^{\bullet}_h$
2. Deletion of edge in (L_i, s_i): we obtain two graphs, one of them contains s_i, (L'_i, s_i) in $\mathcal{T}^{\bullet}_h$, and for the second one containing u (without loss of generality) (L'_i, u) in $\mathcal{T}^{\bullet}_h$.
3. Contraction of edge $s_i s_{i+1}$: we obtain only shorter spine, thus $G/s_i s_{i+1}$ in $\mathcal{C}^{\sqcup}_{h,t-1} \subseteq \mathcal{C}^{\sqcup}_{h,t}$.
4. Deletion of edge $s_i s_{i+1}$: we obtain two components, each is of them is again of a shorter spine.
5. Vertex deletions are analogous.

□

The following lemma is a consequence of Roberson [2022, Lemma 5.14].

Lemma B.11. *It holds* $\mathsf{hom}(\mathcal{C}^{\sqcup}_0, -) \subsetneq \mathsf{hom}(\mathcal{C}^{\sqcup}_1, -)$.

Proof. Follows from a stronger statement about the homomorphism distinguishability closedness of some classes closed on minors and disjoint unions. Specifically for $\mathcal{C}^{\sqcup}_0$, this was shown by Roberson [2022, Lemma 5.14 as remarked in Section 5.1]. For illustration, we give a concrete separating example in Figure 11. □

The following lemma is a consequence of Schindling [2025, Theorem 4.13.].

Lemma B.12. *It holds* $\mathsf{hom}(\mathcal{C}^{\sqcup}_1, -) \subsetneq \mathsf{hom}(\mathcal{C}^{\sqcup}_2, -)$.

Proof. The class of unions of caterpillars $\mathcal{C}^{\sqcup}_1$ corresponds to the graph class of pathwidth at most 1 [Proskurowski and Telle, 1999, Section 6]. We apply a stronger statement about the homomorphism distinguishability closedness of $\mathcal{C}^{\sqcup}_1$, this was shown recently by [Schindling, 2025, Theorem 4.13.]. □

Lemma B.13. *It holds* $\mathsf{hom}(\mathcal{C}_t, -) \subsetneq \mathsf{hom}(\mathcal{C}_{t+1}, -) \subsetneq \mathsf{hom}(\mathcal{T}, -)$ *for every* t in $\mathbb{N}$.

Proof. Case $t \in \{0, 1\}$: For a disjoint union of two graphs F_1, F_2 it holds $\mathsf{hom}(F_1 \sqcup F_2, G) = \mathsf{hom}(F_1, G) \cdot \mathsf{hom}(F_2, G)$, e.g. [Lovász, 2012, pg. 74]. For any h in $\mathbb{N}$, if $\mathsf{hom}(\mathcal{C}^{\sqcup}_h, G)[F_1 \sqcup F_2] \neq \mathsf{hom}(\mathcal{C}^{\sqcup}_h, H)[F_1 \sqcup F_2]$, then $\mathsf{hom}(\mathcal{C}^{\sqcup}_h, G)[F_1] \neq \mathsf{hom}(\mathcal{C}^{\sqcup}_h, H)[F_1]$ or $\mathsf{hom}(\mathcal{C}^{\sqcup}_h, G)[F_2] \neq \mathsf{hom}(\mathcal{C}^{\sqcup}_h, H)[F_2]$. Therefore, we get $\mathsf{hom}(\mathcal{C}^{\sqcup}_h, -) \equiv \mathsf{hom}(\mathcal{C}_h, -)$. Finally, we use Lemma B.11 and Lemma B.12.

General case $t \geqslant 2$: Similarly, the remaining equivalences are due to Neuen and Seppelt [2026][Theorem 1.5], stating a general result for all forests closed on (topological) minors and disjoint unions. □

B.3 Expressivity hierarchy

Corollary B.14. *There is the following expressivity hierarchy such that for any integer $h \geqslant 3$:*

$$\begin{array}{ccccccccccc}
\mathsf{hom}(\mathcal{P},-) & \subsetneq & \mathsf{hom}(\mathcal{C}_1,-) & \subsetneq & \mathsf{hom}(\mathcal{C}_2,-) & \subsetneq \cdots \subsetneq & \mathsf{hom}(\mathcal{C}_h,-) & \subsetneq \cdots \subsetneq & \mathsf{hom}(\mathcal{T},-) \\
\rotatebox{90}{$=$} & & \rotatebox{90}{$=$} & & \rotatebox{90}{$=$} & & \rotatebox{90}{$=$} & & \rotatebox{90}{$=$} \\
\mathcal{I}_R(-,\chi_{\mathrm{triv}}) & \subsetneq & \mathcal{I}_R(-,\chi_{\mathrm{deg}}) & \subsetneq & \mathcal{I}_R(-,\chi_{\mathsf{cr}}^{(2)}) & \subsetneq \cdots \subsetneq & \mathcal{I}_R(-,\chi_{\mathsf{cr}}^{(h)}) & \subsetneq \cdots \subsetneq & \mathsf{cr}(-),
\end{array}$$

Proof. The second equivalence in the hierarchy is given by Theorem B.4, while the intermediate equivalences follow from Theorem 4.1 and Theorem 4.2. The strict separations follow from Lemma B.13. The last equivalence follows from Theorem B.6. □

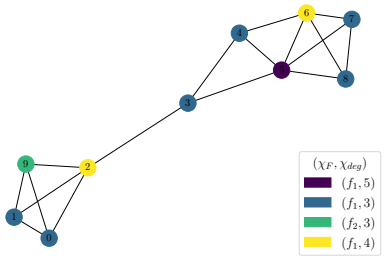

Figure 12: An example graph representation of a protein structure, colored by $(\chi_F, \chi_{\mathrm{deg}})$. The shown datapoint is taken from the Proteins dataset [Morris et al., 2020].

C Graph Neural Networks

In this section, we further discuss variants of Caterpillar GNN, a practical architecture built upon the theoretical foundations of graph-level aggregation. Graphs in real-world datasets such as Proteins [Morris et al., 2020], ZINC [Irwin et al., 2012], or ESOL [Wu et al., 2018] typically come attributed with *vertex features*. An example is given in Figure 12. Here, we assume that these features are seen as categorical, taken from a finite set Σ. Rather than encoding continuous-valued properties, these features represent discrete properties such as an atom type or molecular class.

The vertex features of a given graph G, we represent as a coloring function $\chi_F(G,-)\colon V(G) \to \Sigma$. To seamlessly integrate vertex features with our scalable aggregation scheme, we introduce a parametrized combined coloring that incorporates both the vertex features and the refinement coloring:

$$\tilde{\chi}^{(h)}(G,u) = (\chi_F(G,u), \chi_{\mathsf{cr}}^{(h)}(G,u)), \tag{27}$$

for every vertex u in $V(G)$ and h in $\mathbb{N}$. The primary motivation for combining with χ_F is to prevent the computational graph from being too downscaled to accommodate distinct vertex features. Technically, in the following architecture, we employ vertex-feature matrices $\boldsymbol{Y} = \boldsymbol{Y}(G,\chi_F)$ indexed by (specifically selected) colored walks $\boldsymbol{a}c$ in Σ^{t+1}, and feature channels i. Crucially, each entry $Y[\boldsymbol{a}c,i]$ only needs to represent the i-th channel of the vertex feature $c = \chi_F(G,u)$ for a suitable u in $V(G)$. This is due to the structure of the prefix-successor relation we explained in Section 3.

C.1 Caterpillar GCN

Let L denote the number of layers in the network. For each layer ℓ, where $0 \leqslant \ell \leqslant L$ and $t_\ell = L - \ell$, we define the following parameters: c_ℓ in $\mathbb{N}$, the number of feature channels; $\boldsymbol{W}^{(\ell)}$ in $\mathbb{R}^{c_\ell \times c_{\ell+1}}$, a learnable weight matrix; $\boldsymbol{Y}^{(\ell)}$, vertex feature matrix with channel embeddings for c_ℓ; σ, an activation function, e.g. ReLU. Recall from Section 3 that $S_t = S_t(G,\tilde{\chi}) \subseteq \Sigma^t$ denotes canonical subsets of colored walks of size at most $|V(G)|$.

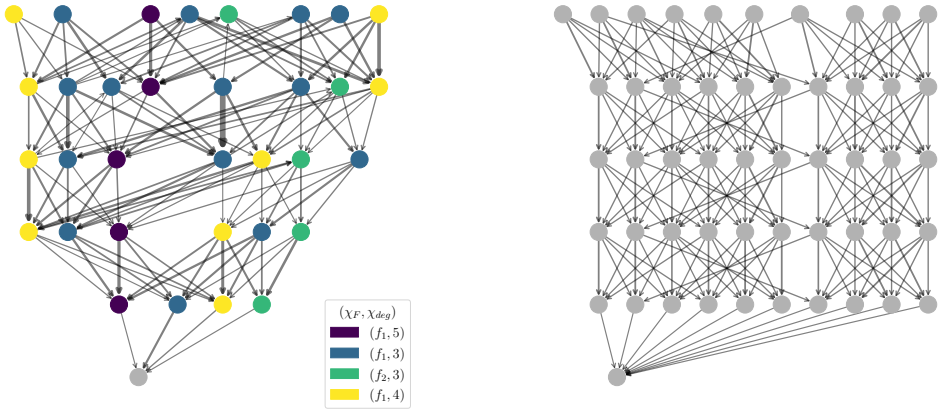

Figure 13: Left: Computational graph **(ours)** of Caterpillar GCN with $h = 1$ that uses $\boldsymbol{C}_{t_\ell}^{\tilde{\boldsymbol{A}}}$ matrix at ℓ-th layer. This diagram is analogous to Figure 4. Right: Computational graph (message-passing) of GCN that uses copies of matrix $\tilde{\boldsymbol{A}}$ for its layers. Weights are represented by line width, and signs by arrow direction. The input graph is shown in Figure 12.

We derive a *Caterpillar GCN* as a sequence of layers transforming feature matrixes. Specifically, for each layer ℓ, we define the features $\boldsymbol{h}^{(\ell)}$ in $\mathbb{R}^{S_{t_\ell} \times c_\ell}$ as follows:

$$\boldsymbol{h}^{(0)} = \boldsymbol{Y}^{(0)}, \quad \boldsymbol{h}^{(\ell+1)} = \sigma\big(\boldsymbol{C}_{t_\ell}^{\tilde{\boldsymbol{A}}}\, \boldsymbol{h}^{(\ell)}\, \boldsymbol{W}^{(\ell)}\big) \square\, \boldsymbol{Y}^{(\ell)}, \quad \boldsymbol{h}^{(L)} = \boldsymbol{C}_0^{\boldsymbol{I}}\, \boldsymbol{h}^{(L)}\, \boldsymbol{W}^{(L)},$$

where $\square$ represents a standard addition or a concatenation. Finally, the resulting graph-level feature is

$$\boldsymbol{h}(L, \theta; G, \chi) \coloneqq \boldsymbol{h}^{(L)} \text{ in } \mathbb{R}^{\{\lambda\} \times c_L}, \tag{28}$$

where θ is the set of all learnable parameters. For the ℓ-th layer, we used the t_ℓ-th reduced matrix (Equation 2). We specifically set $\boldsymbol{M} \coloneqq \tilde{\boldsymbol{A}}$ for the reduced variant $\boldsymbol{C}_t^{\boldsymbol{M}}$, where $\tilde{\boldsymbol{A}}$ is the augmented normalized adjacency matrix, see Kipf and Welling [2017], defined as follows: $\tilde{\boldsymbol{A}} = \hat{\boldsymbol{D}}^{-1/2} \hat{\boldsymbol{A}} \hat{\boldsymbol{D}}^{-1/2}$, where $\hat{\boldsymbol{A}} = \boldsymbol{A} + 2\boldsymbol{I}$, and $\hat{\boldsymbol{D}} + 2\boldsymbol{I}$. For an example of such a computational graph, see Figure 13.

Indexing of the reduced variants of graph matrices ensures the correct structure of aggregation from longer to shorter words, and, ultimately, $\boldsymbol{C}_0^{\boldsymbol{I}}$ maps to the space of dimension $S_0 = \{\lambda\}$, as shown in Figure 4. Note that if we set $\tilde{\chi} = (\chi_F, \chi_{\mathrm{id}})$ and operation $\square$ to ignore $\boldsymbol{Y}$, our architecture becomes nothing else than a network of GCNConv [Kipf and Welling, 2017] with global readout.

C.2 Density of Computational Graphs

As defined in the previous subsection, while computational graphs of GCNs are given by matrix $\tilde{\boldsymbol{A}}$ the computational graphs of Caterpillar GCNs are given by the reduced variants $\boldsymbol{C}_t^{\tilde{\boldsymbol{A}}}$ (see Figure 4 and Figure 13).

Even though the number of nodes of the computational graph may *decrease* significantly with height, as shown in Figure 15, and further detailed in Table 6, this does not necessarily imply a proportional decrease of the number of edges. To study this phenomenon, we introduce two metrics to quantify the sparsity of computational graphs.

For any matrix $\boldsymbol{M}$ in $\mathbb{R}^{R \times C}$, we denote the number of its *non-zero entries* by

$$\mathrm{nnz}(\boldsymbol{M}) \coloneqq \left|\{(r, c) \mid \boldsymbol{M}[r, c] \neq 0,\ r \text{ in } R,\ c \text{ in } C\}\right|.$$

Then the *density* of $\boldsymbol{M}$ is defined as the ratio of its non-zero entries to the total number of entries, i.e., $\mathrm{nnz}(\boldsymbol{M}) \cdot |R \times C|^{-1}$. Complementarily, the *sparsity* of $\boldsymbol{M}$ is defined as $1 - \mathrm{nnz}(\boldsymbol{M}) \cdot |R \times C|^{-1}$.

Absolute density. We extend density to the computational graph of Caterpillar GCN by taking the average of layer densities. Let G be an input graph with a Σ-coloring χ, and let L denote the number

of layers.

$$\text{absolute_density}(L, G, \chi) \coloneqq \frac{1}{L} \sum_{t=1}^{L} \frac{\text{nnz}\left(\boldsymbol{C}_t^{\tilde{\boldsymbol{A}}}\right)}{|S_t \times S_{t+1}|}. \tag{29}$$

Relative density. By relative density $\delta_{h,L}$, we measure the ratio of the number of edges in the computational graph of Caterpillar GCN to that of (standard) GCN.

$$\text{relative_density}(L, G, \chi) \coloneqq \frac{1}{L} \sum_{t=1}^{L} \frac{\text{nnz}\left(\boldsymbol{C}_t^{\tilde{\boldsymbol{A}}}\right)}{\text{nnz}\left(\boldsymbol{A}\right)}, \quad \text{and}$$
$$\delta_{h,L} \coloneqq \text{relative_density}\left(L, G, (\chi_F, \chi_{\mathsf{cr}}^{(h)})\right). \tag{30}$$

Note that for the computational graph of GCN, the relative density equals one. By Proposition 3.3, we can state this formally as $\text{relative_density}(L, G, \chi_{\text{id}}) = 1.0$. Moreover, since $\text{nnz}(\boldsymbol{C}_t^{\tilde{\boldsymbol{A}}}) = \text{nnz}(\boldsymbol{C}_t^{\hat{\boldsymbol{A}}}) = \text{nnz}(\boldsymbol{C}_t^{\boldsymbol{A}}) + \text{nnz}(\boldsymbol{C}_t^{\boldsymbol{I}})$, these metrics extend naturally to computational graphs of other variants of GNNs.

Densities and dimensions of $\boldsymbol{C}_t^{\tilde{\boldsymbol{A}}}$ on real-world datasets. To better understand the sparsity (density) of computational graphs induced by our model, we empirically evaluate how the density evolves with increasing height parameter h for fixed $L = 5$. Specifically, for a fixed sequence length $L = 5$ and varying height parameter h, we measure the densities (defined above) of the resulting computational graphs across TUDataset, MoleculeNet, and ZINC, as summarised in Table 2.

In our experiments, we observe that the relative density $\delta_{h,5}$ is, in most cases, strictly below 1.0, and can overall be bounded by a small constant, for instance $\delta_{h,5} \lessapprox 2.0$. This implies that the matrices $\boldsymbol{C}_t^{\tilde{\boldsymbol{A}}}$ *typically contain fewer non-zero entries* than the original adjacency matrix $\tilde{\boldsymbol{A}}$. On unattributed graphs such as IMDB-BINARY and COLLAB, the ratio $\delta_{h,5}$ tends to be even lower (e.g., approximately 0.1 and 0.3, respectively).

On the other hand, the absolute_density tends to increase as the height parameter h decreases. This indicates that the computational graphs become significantly *denser*, which is beneficial for execution on hardware accelerators, provided that the relative density $\delta_{h,L}$ does not increase by more than a small constant factor. On typically sparse datasets such as PROTEINS and ENZYMES, we observed a substantial rise in absolute density (from approximately 20% for MP to around 50% for C_0), suggesting that such computational graphs may be more desirably processed using dense tensor operations rather than sparse ones.

C.3 Impacts of Relative Density on Asymptotic Complexity

In this subsection, we discuss the asymptotic time complexity of Caterpillar GNN (including precomputation) in comparison to some other methods learning on sequential patterns in graphs. We first focus on relative density $\delta_{h,L}$, and later, we also discuss the impact of the number of nodes in the computational graph.

When the number of channels d in the GNN is taken as a constant, we give a short overview Section C.3. Assuming that an input graph G on n vertices is connected and $n \geqslant 3$, it holds for the maximal degree Δ that $2 \leqslant \Delta \leqslant n - 1$. A multiplicative factor in our comparison $\delta_{h,L}$ is the average relative density, as defined in Section C.2. For Caterpillar GNNs, it is fixed that $L = T$. The best known theoretical bound to our knowledge is the trivial one: $\delta_{h,L} \leqslant n^2/|E|$. Importantly, on TUDataset, MoleculeNet and ZINC, we see that $\delta_{h,5}$ is typically a fraction smaller than 1.0 and overall $\delta_{h,5} \lessapprox 2.0$, see Table 2.

Complexity under large width. In the regime where the number of channels d (i.e., the width) of our GNN is large, the multiplication of learnable parameters $\boldsymbol{W}^{(\ell)}$ in $\mathbb{R}^{d \times d}$ with features $\boldsymbol{h}^{(\ell)}$ in $\mathbb{R}^{|S_{t_\ell}| \times d}$ (see Section C.1) becomes a dominant contributor to both time and space complexity. Concretely, the cost of one layer is

$$\mathcal{O}\big(\text{nnz}(\boldsymbol{C}_{t_\ell}^{A}) \cdot d + |S_{t_\ell+1}| \cdot d^2\big), \tag{31}$$

type	MUTAG		PROTEINS		ENZYMES		IMDB-BINARY		COLLAB	
	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]
C_0	0.73 ± 0.63	44.4 ± 10.5	1.26 ± 1.80	53.9 ± 29.4	1.46 ± 1.68	48.4 ± 26.1	0.01 ± 0.00	100.0 ± 0.0	0.35 ± 0.39	51.4 ± 24.7
C_1	1.23 ± 0.82	36.8 ± 10.9	1.59 ± 1.69	38.2 ± 22.6	1.40 ± 0.77	31.4 ± 15.0	0.11 ± 0.10	73.4 ± 16.3	0.35 ± 0.39	51.4 ± 24.7
C_2	0.94 ± 0.33	27.5 ± 6.9	0.81 ± 0.24	27.5 ± 23.9	0.84 ± 0.20	20.8 ± 13.0	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.8 ± 26.2
C_3	0.85 ± 0.22	24.4 ± 7.0	0.78 ± 0.21	27.0 ± 24.1	0.82 ± 0.17	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_4	0.83 ± 0.20	23.6 ± 7.3	0.78 ± 0.21	26.9 ± 24.1	0.82 ± 0.17	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_5	0.83 ± 0.19	23.4 ± 7.4	0.78 ± 0.21	26.9 ± 24.1	0.82 ± 0.16	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_6	0.83 ± 0.19	23.4 ± 7.4	0.78 ± 0.21	26.9 ± 24.1	0.82 ± 0.16	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_7	0.83 ± 0.19	23.4 ± 7.4	0.78 ± 0.21	26.9 ± 24.1	0.82 ± 0.16	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_8	0.83 ± 0.19	23.4 ± 7.4	0.78 ± 0.21	26.9 ± 24.1	0.82 ± 0.16	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_9	0.83 ± 0.19	23.4 ± 7.4	0.78 ± 0.21	26.9 ± 24.1	0.83 ± 0.16	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
C_{10}	0.83 ± 0.19	23.4 ± 7.4	0.78 ± 0.21	26.9 ± 24.1	0.83 ± 0.16	20.4 ± 13.1	0.10 ± 0.09	72.0 ± 17.7	0.30 ± 0.29	50.9 ± 26.2
MP	1.00 ± 0.00	18.9 ± 4.7	1.00 ± 0.00	24.4 ± 20.6	1.00 ± 0.00	19.0 ± 11.8	1.00 ± 0.00	54.7 ± 23.4	1.00 ± 0.00	51.8 ± 29.9

type	FreeSolv		Lipo		ESOL		BACE		ZINC	
	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]	$\delta_{h,5}$ [↓]	absolute [↑, %]
C_0	0.80 ± 0.34	51.0 ± 20.5	1.13 ± 0.39	18.9 ± 6.9	0.88 ± 0.44	40.3 ± 19.8	1.09 ± 0.32	14.1 ± 4.5	1.33 ± 0.50	23.0 ± 6.1
C_1	0.80 ± 0.34	51.0 ± 20.5	1.13 ± 0.39	18.9 ± 6.9	0.88 ± 0.44	40.3 ± 19.8	1.09 ± 0.32	14.1 ± 4.5	1.29 ± 0.47	22.4 ± 5.9
C_2	0.77 ± 0.27	49.4 ± 21.1	0.93 ± 0.17	15.4 ± 5.4	0.81 ± 0.28	37.6 ± 20.0	0.97 ± 0.15	11.9 ± 3.3	0.99 ± 0.17	16.9 ± 3.6
C_3	0.75 ± 0.25	48.9 ± 21.5	0.88 ± 0.12	14.5 ± 5.2	0.78 ± 0.25	36.6 ± 20.4	0.90 ± 0.08	11.0 ± 3.0	0.91 ± 0.10	15.5 ± 3.3
C_4	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.3 ± 5.1	0.78 ± 0.24	36.5 ± 20.5	0.89 ± 0.07	10.9 ± 3.0	0.90 ± 0.09	15.3 ± 3.3
C_5	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.2 ± 5.1	0.78 ± 0.24	36.5 ± 20.5	0.89 ± 0.07	10.9 ± 3.0	0.90 ± 0.09	15.3 ± 3.3
C_6	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.2 ± 5.1	0.78 ± 0.24	36.5 ± 20.5	0.89 ± 0.07	10.9 ± 3.0	0.90 ± 0.09	15.2 ± 3.3
C_7	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.2 ± 5.1	0.78 ± 0.24	36.5 ± 20.5	0.89 ± 0.07	10.8 ± 3.0	0.90 ± 0.09	15.2 ± 3.3
C_8	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.2 ± 5.1	0.78 ± 0.24	36.5 ± 20.5	0.89 ± 0.07	10.9 ± 3.0	0.90 ± 0.09	15.2 ± 3.3
C_9	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.2 ± 5.1	0.78 ± 0.24	36.4 ± 20.5	0.89 ± 0.07	10.9 ± 3.0	0.90 ± 0.09	15.2 ± 3.3
C_{10}	0.75 ± 0.25	48.9 ± 21.6	0.87 ± 0.11	14.2 ± 5.1	0.78 ± 0.24	36.4 ± 20.5	0.89 ± 0.07	10.9 ± 3.0	0.90 ± 0.09	15.2 ± 3.3
MP	1.00 ± 0.00	39.0 ± 16.3	1.00 ± 0.00	12.8 ± 4.3	1.00 ± 0.00	28.9 ± 14.9	1.00 ± 0.00	9.9 ± 2.8	1.00 ± 0.00	14.1 ± 2.9

Table 2: *Density of computational graphs across datasets.* By C_h is denoted the (caterpillar height) parameter h in $\mathbb{N}$ of graph-level aggregation (ours), while MP denotes the full message-passing GCN. The subcolumns correspond to $\delta_{h,L}$ (lower is better) and $\mathrm{relative_density}$ (higher is better once $\delta_{h,L}$ is bounded) for $L = 5$ layers. We report absolute density in $\%$.

Method	Pre-comp.	Time per layer	Worst case	Equivariant
MPGNN [Gilmer et al., 2017]	—	$\lvert E\rvert$	n^2	✓
CRaWL [Tönshoff et al., 2023]	—	$n \cdot f_G(\Delta^T) \cdot T$	$n f_G((n-1)^T)\, T$	X
NeuralWalker [Chen et al., 2024]	—	$n \cdot f_G(\Delta^T) \cdot T$	$n f_G((n-1)^T)\, T$	X
Path NN [Michel et al., 2023]	$n \cdot \Delta^T$	$n \cdot \Delta^T \cdot T$	$n^{T+1}\, T$	✓
Path-based GNN [Graziani et al., 2024]	$n \cdot \Delta^T$	$n \cdot \Delta^T \cdot T$	$n^{T+1}\, T$	✓
Caterpillar GNN (ours)	$T \cdot n^{2.372}$	$\lvert E\rvert \cdot \delta_{h,L}$	n^2	✓

Table 3: Asymptotic comparison of methods processing sequential patterns of length T. The function f_G gives the number of sampled walks on $G = (V, E)$ from the given upper bound. Symbol Δ stands for the maximal degree and $\delta_{h,L}$ is the relative density on G (see Equation 30).

hence for a network with L layers ($t_\ell = L - t$) and sufficiently large d, the term $d^2 \cdot \sum_{t=0}^{L} |S_t|$ dominates the overall complexity. Hence, the *number of nodes in the computational graph*, $\sum_{t=0}^{L} |S_t|$, appears as a multiplicative factor in the time and memory complexity and is asymptotically significant with respect to the width d.

Empirically, on real-world datasets TUDataset, MoleculeNet, and ZINC, we observe that the number of nodes in the computational graph tends to *decrease* as the height parameter h decreases, as illustrated in Figure 15 and detailed further in Tables 6 and 8.

C.4 Towards Modern Architectures and Large Graph-Level Datasets

To broaden our empirical evaluation, this subsection extends our analysis of computational graphs to more practical settings (Table 4). In the previous analysis, we fixed as many hyperparameters as possible, such as the number of layers. Here, in contrast, for structural comparison against a strong message-passing baseline, we adopt the experimental settings of the recent GCN-based architecture (GCN$^+$) proposed by Luo et al. [2025]. Their work demonstrates that, with careful design and tuning, message-passing backbones can be highly competitive on graph-level prediction tasks.

Following their setup, we preserve the configuration parameters such as the number of layers, and the dimensionality of the preprocessed node features, and we reuse their protocol to ensure a controlled structural comparison where only the aggregation mechanism differs. To further isolate

	MalNet-Tiny features: 5 graphs: 5,000 avg. $\lvert V\rvert$: 1410.3 avg. $\lvert E\rvert$: 2859.9 layers: $L = 8$ (GCN$^+$)					**Peptides-func** features: 9 graphs: 15,535 avg. $\lvert V\rvert$: 150.9 avg. $\lvert E\rvert$: 307.3 layers: $L = 3$ (GCN$^+$)					**Peptides-struct** features: 9 graphs: 15,535 avg. $\lvert V\rvert$: 150.9 avg. $\lvert E\rvert$: 307.3 layers: $L = 5$ (GCN$^+$)				
	edge density		*nodal efficiency*		*precomp.*	*edge density*		*nodal efficiency*		*precomp.*	*edge density*		*nodal efficiency*		*precomp.*
type	$\delta_{h,8}$[↓]	abs.[↑, %]	#n.[↓]	%s.[↑]	time[↓]	$\delta_{h,3}$[↓]	abs.[↑, %]	#n.[↓]	%s.[↑]	time[↓]	$\delta_{h,5}$[↓]	abs.[↑, %]	#n.[↓]	%s.[↑]	time[↓]
C_0	30.52	24.7	5971.2	53.0	6.97h	1.634	15.4	233.7	61.4	8.2min	2.116	12.4	471.5	48.0	13.1min
C_1	2.002	7.7	6757.5	46.8	7.85h	1.634	15.4	233.7	61.4	9.9min	2.116	12.4	471.5	48.0	15.0min
C_2	1.500	6.4	7436.9	41.4	9.63h	1.418	8.7	292.3	51.7	17.0min	1.708	7.4	542.2	40.2	22.0min
C_3	1.516	6.3	7900.7	37.8	10.39h	1.416	5.9	366.1	39.5	24.4min	1.533	5.3	629.0	30.6	34.3min
C_4	1.519	6.3	7953.9	37.3	11.32h	1.344	4.5	429.3	29.0	36.5min	1.412	4.2	698.9	22.9	45.4min
C_5	1.519	6.3	7959.8	37.3	10.93h	1.350	3.9	478.3	20.9	50.9min	1.393	3.8	754.0	16.8	58.4min
C_6	1.519	6.3	7960.7	37.3	10.94h	1.372	3.6	515.9	14.7	54.4min	1.398	3.5	797.1	12.1	1.15h
MP	1.000	0.8	12693.8	0.0	–	1.000	2.1	604.8	0.0	–	1.000	2.1	906.6	0.0	–

	ogbg-code2 features: 2 graphs: 452,741 avg. $\lvert V\rvert$: 121.6 avg. $\lvert E\rvert$: 333.9 layers: $L = 4$ (GCN$^+$)					**ogbg-molhiv** features: 9 graphs: 41,127 avg. $\lvert V\rvert$: 25.5 avg. $\lvert E\rvert$: 54.9 layers: $L = 4$ (GCN$^+$)					**ogbg-molpcba** features: 9 graphs: 437,929 avg. $\lvert V\rvert$: 26.0 avg. $\lvert E\rvert$: 56.2 layers: $L = 10$ (GCN$^+$)				
	edge density		*nodal efficiency*		*precomp.*	*edge density*		*nodal efficiency*		*precomp.*	*edge density*		*nodal efficiency*		*precomp.*
type	$\delta_{h,4}$[↓]	abs.[↑, %]	#n.[↓]	%s.[↑]	time[↓]	$\delta_{h,4}$[↓]	abs.[↑, %]	#n.[↓]	%s.[↑]	time[↓]	$\delta_{h,10}$[↓]	abs.[↑, %]	#n.[↓]	%s.[↑]	time[↓]
C_0	1.638	9.4	445.2	26.9	10.21h	1.467	28.7	84.2	34.5	10.7min	1.758	22.0	235.3	17.9	4.46h
C_1	1.476	8.3	454.8	25.3	12.09h	1.467	28.7	84.2	34.5	11.4min	1.758	22.0	235.3	17.9	4.57h
C_2	1.318	6.9	486.6	20.1	16.14h	1.223	21.1	92.6	27.9	16.1min	1.365	16.2	245.0	14.5	6.32h
C_3	1.274	6.1	526.0	13.6	22.11h	1.188	18.7	98.9	23.1	20.4min	1.291	14.7	251.9	12.1	7.97h
C_4	1.293	5.7	571.7	6.1	56.26h	1.190	18.1	101.7	20.9	22.5min	1.283	14.4	254.5	11.2	8.78h
C_5	1.299	5.6	589.0	3.3	61.11h	1.192	18.0	102.8	20.0	24.3min	1.282	14.4	255.3	10.9	9.10h
C_6	1.300	5.6	595.5	2.2	64.88h	1.194	17.9	103.4	19.6	49.9min	1.282	14.4	255.6	10.9	8.66h
MP	1.000	4.3	608.9	0.0	–	1.000	9.8	128.6	0.0	–	1.000	8.8	286.7	0.0	–

Table 4: *Density and nodal efficiency of computational graphs across large datasets.* By C_h is denoted the (caterpillar height) parameter h in $\mathbb{N}$ of graph-level aggregation (ours), while MP denotes the full message-passing GCN$^+$ [Luo et al., 2025]. The subcolumns correspond to $\delta_{h,L}$ (lower is better) and relative_density (higher is better once $\delta_{h,L}$ is bounded) for $L = 5$ layers. We report absolute density in %. Columns "#n." denote the number of *nodes* of the computation graph, and columns "%s." percentage of nodes of the computation graph *saved* compared with message-passing (MP).

the impact of lower expressivity, we intentionally disable positional encodings and edge features for all architectures, consistently with the GCN$^+$ ablation setting used in Table 6 of Section 5.2 in [Luo et al., 2025]. Hence, we obtain a clear assessment of how graph-level aggregation contributes to graph-level performance uninfluenced by other structural augmentations.

We analyze the computational graphs of the MalNet-Tiny dataset [Freitas et al., 2021], which contain graphs with up to $5,000$ nodes ($1,410$ on average). In addition, we also include an analysis of datasets with discrete node features: Peptides-func, Peptides-struct from Open Graph Benchmark Hu et al. [2021], and datasets ogbg-code2, ogbg-molhiv, ogbg-molpcba from Long-Range Graph Benchmark [Dwivedi et al., 2022], see Table 4. These datasets allow us to examine how the structure of proposed inductive biases behaves under substantially larger graph structures and more diverse data modalities. We report the preprocessing times of a *serial* precomputation on the platform of an Intel Xeon Platinum 8168 @ 2.70 GHz processor (24 cores, 48 threads, 33 MB L3 cache, 64B cache line).

Recipe for equivariant density reduction. Our proposed reduced variants of graph matrices are fully (permutation) equivariant. In practice, we can therefore further reduce the value of relative density $\delta_{h,L}$ (Equation 30) by, e.g., thresholding small-magnitude entries of our matrices. This optional step decreases computational cost while preserving the symmetries inherent to our construction. On the other hand, during our measurements in Table 4, similarly to our previous analysis, we observe that the relative density $\delta_{h,L}$ of the computational graphs is a very small constant. The only exception we found was in the case of $\delta_{h,8}$ for the MalNet-Tiny and the height $h = 0$, for which we paid particular attention to eliminate possible hardware errors.

D Details of Experiments

D.1 Additional Conclusions for Paradoxically Reversed Descent

We observed a double descent, which we attribute to training oscillation between two regimes: digit-by-digit processing, and a higher-level aggregation, producing the high-variance performance

dip. As we scale our models (cf. Section 3.3, Section 4), incidence topology scales analogously from $\tau(\chi_{\text{triv}})$ up to $\tau(\chi_{\text{id}}) = \tau(E)$. Unlike rewiring strategies, e.g., Topping et al. [2022], changing edges E' to operate in $\tau(E')$, our approach restructures the computational graph into a less expressive one (Section 4). Finally, given the systematically decreasing performance of models C_{10} up to MP, a bottleneck of *information alignment* of $\tau(E)$ is revealed by the topology $\tau(\chi_{\deg})$ that qualitatively differs from, e.g., oversquashing [Alon and Yahav, 2021].

D.2 Incidence Topology

Although our discussion of topology mainly serves heuristic and interpretative purposes, we provide formal definitions of the relevant notions in Section 5.1.

Topology. Let V be a set and $\tau \subseteq 2^V$ be a family of subsets of V. Then τ is a *topology* on V if the following three conditions hold:

(T1) Set V in τ, and empty set $\varnothing$ in τ.

(T2) The family τ is closed on all unions of its sets.

(T3) The family τ is closed on finite intersections of its sets.

A subfamily $\mathcal{B} \subseteq \tau$ is a *subbase* for τ if τ is the intersection of all topologies on V containing $\mathcal{B}$. We say that $\mathcal{B}$ *generates* τ.

For a graph $G = (V, E)$ on V, the neighborhood topology $\tau(E)$ is generated by the following family of sets for every v in V and r in $\mathbb{N}$ such that $r > 0$:

$$B_r(v) = \{u \mid u \text{ in } V,\ (\boldsymbol{I} + \boldsymbol{A})^r[u, v] \neq 0\},$$

where $\boldsymbol{A}$ is the adjacency and $\boldsymbol{I}$ is the (self-loop) identity matrix of shape $V \times V$ for G.

Let χ be a Σ-coloring on G and let T be the maximum length of colored walks. Then we denote the following family of sets for every $\boldsymbol{a}$ in $\Sigma^{\leqslant T}$

$$B(\boldsymbol{a}, T) \coloneqq \{u \mid u \text{ in } V,\ (\boldsymbol{I} + \boldsymbol{A})\boldsymbol{W}[u, \boldsymbol{a}] \neq 0\}. \tag{32}$$

where we recall the walk-incidence matrix $\boldsymbol{W}$ of shape $V \times \Sigma^*$ defined in Section 3.1.

We call the topology generated by the subbase $\mathcal{B} = \{B(\boldsymbol{a}, T) \mid \boldsymbol{a} \text{ in } \Sigma^{\leqslant T}\}$ the *incidence topology* $\tau(\chi)$ on V. We state the following proposition to highlight the similarity with Observation 3.2 and Proposition 3.3.

Proposition D.1. *Let $G = (V, E)$ be a graph without isolated vertices, and $T > 0$. Then the topology $\tau(\chi_{\text{triv}})$ is trivial, and the topology $\tau(\chi_{\text{id}}) = \tau(E)$.*

Proof. Let us denote the colored walk of a single color of length t by $\boldsymbol{z}_t$ (as in Observation 3.2). Since every node u in V has at least one neighbor, it is adjacent to colored walk $\boldsymbol{z}_{2t}$ and incident to colored walk $\boldsymbol{z}_{2t+1}$. That is, $(\boldsymbol{AW})[u, \boldsymbol{z}_{2t}] \neq 0$ and $(\boldsymbol{IW})[u, \boldsymbol{z}_{2t+1}] \neq 0$ and thus $\tau(\chi_{\text{triv}})$ contains only $\varnothing$, and V. Therefore, the subbase is of $B(\boldsymbol{z}_t, T) = V$, and thus $\tau(\chi_{\text{triv}}) = \{\varnothing, V\}$.

For the second part, we recall that χ_{id} is a V-coloring. For v in V we take any colored walk $\boldsymbol{a}v$, and we get $(\boldsymbol{I} + \boldsymbol{A})W[u, \boldsymbol{a}v] = (\boldsymbol{I} + \boldsymbol{A})[u, v]$. Finally, every ball $B_r(v)$ in $\tau(E)$ is (already) generated by the union of some 1-balls of the form $B_1(v) = B(\boldsymbol{a}v, T) = \{u \mid u \text{ in } V,\ (\boldsymbol{I} + \boldsymbol{A})[u, v] \neq 0\}$. □

D.3 Generation of NSTEPADDITION Dataset

We construct the dataset for a binary classification task, where each graph encodes two integers represented in binary. For an example of a graph G encoding two integers 11 and 22 in the 5-bit binary case, see Figure 14. To construct a graph of our dataset of a target sum $N = 2^{B-1}$, we randomly generate two integers x_1 and x_2 in the range $[0, 2^B - 1]$, such that $x_1 + x_2 = N$. With 0.5 probability, we add an offset to one of the numbers in the range of $[0, \frac{2}{3}N]$ to deviate from the target sum N by not more than 66.7%. Accordingly, we assign the class of the graph to be 1 if $x_1 + x_2 = N$, and 0 otherwise. We do not use any node features, so the classification relies solely on graph structure. Samples: 6,000 graphs with integers generated using $B = 15$ bits. The dataset we provide is balanced with respect to the class labels.

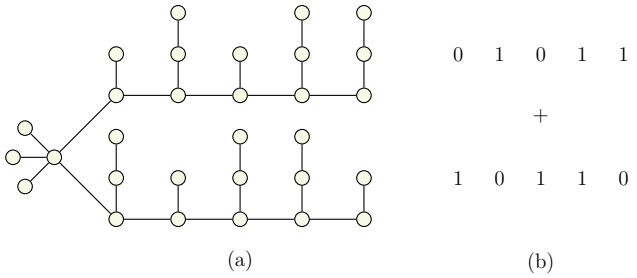

Figure 14: NSTEPADDITION: An example of a graph G (a). The graph G encodes two 5-bit numbers (b), namely 11 (top) and 22 (bottom). If $N = 33$ for a classification dataset then G is a positive example, as $11 + 22 = 33$.

type	epochs $= 10$			epochs $= 50$		
	mean $\pm$ std	#n.	%s.	mean $\pm$ std	#n.	%s.
$\mathcal{C}_0$	0.590 $\pm$ 0.068	18.0	98.7	0.601 $\pm$ 0.124	18.0	98.7
$\mathcal{C}_1$	**0.929 $\pm$ 0.013**	1011.7	25.1	**0.971 $\pm$ 0.010**	1011.7	25.1
$\mathcal{C}_2$	0.688 $\pm$ 0.046	1116.4	17.3	0.869 $\pm$ 0.061	1116.4	17.3
$\mathcal{C}_3$	0.637 $\pm$ 0.084	1189.9	11.9	0.884 $\pm$ 0.056	1189.9	11.9
$\mathcal{C}_4$	0.807 $\pm$ 0.053	1242.6	8.0	0.897 $\pm$ 0.029	1242.6	8.0
$\mathcal{C}_5$	0.839 $\pm$ 0.016	1275.5	5.6	0.887 $\pm$ 0.027	1275.5	5.6
$\mathcal{C}_8$	0.584 $\pm$ 0.045	1299.4	3.8	0.645 $\pm$ 0.023	1299.4	3.8
$\mathcal{C}_{10}$	0.568 $\pm$ 0.025	1299.8	3.8	0.633 $\pm$ 0.014	1299.8	3.8
$\mathcal{C}_{12}$	0.569 $\pm$ 0.021	1299.9	3.7	0.626 $\pm$ 0.022	1299.9	3.7
$\mathcal{C}_{14}$	0.562 $\pm$ 0.016	1299.9	3.7	0.625 $\pm$ 0.013	1299.9	3.7
$\mathcal{C}_{16}$	0.563 $\pm$ 0.024	1299.9	3.7	0.634 $\pm$ 0.021	1299.9	3.7
$\mathcal{C}_{18}$	0.563 $\pm$ 0.024	1299.9	3.7	0.634 $\pm$ 0.021	1299.9	3.7
MP	0.524 $\pm$ 0.011	1350.5	0.0	0.629 $\pm$ 0.102	1350.5	0.0

Table 5: Results for the NSTEPADDITION dataset. By C_h is denoted the (caterpillar height) parameter h in $\mathbb{N}$ of graph-level aggregation (ours), while MP denotes the full message-passing GCN. We report mean validation accuracy with standard deviation across 10 different splits. The 18-layer model was trained for 10 and 50 epochs. The best results are highlighted in bold. The columns "#n." denote the number of *nodes* of the computation graph, and columns "%s." the percentage of nodes of the computation graph *saved* compared with message-passing (MP).

D.4 Implementation Details

We use the PyTorch Geometric library [Paszke et al., 2019] to implement our models. As it might be considered common, we precomputed the normalization of the (sparse) adjacency matrix for the `GCNConv` layer, to use it later in the forward pass with the option `norm=False`, and add the normalized weights to the message-passing instead. For the cases where Caterpillar GCN is not identical to message passing, we added an efficient (sparse) version of the adjacency matrix $C^{\tilde{A}}_{t_\ell}$, specialized for every layer ℓ, in a way similar to the normalization of the adjacency matrix. We preprocessed datasets in a unified fashion for message-passing and Caterpillar GCN (C_0 up to C_{10}).

We trained our models on the NSTEPADDITION dataset described above using deeper architectures (18 layers), with small hidden dimensions (width=8), batch size of 64, and moderate regularization settings (final dropout 0.3, weight decay 10^{-6}). Training was performed for 10 Figure 7 and 50 epochs using 10-fold cross-validation. Complete results are reported in Table 5. We deliberately adopted a minimal configuration in order to cleanly isolate the topological effect of computational graph scaling behind our approach. We remark that the definition of incidence topology in general supports richer benchmarks. The ongoing work may extend these experiments to include multiple occurrences (multiple graph branches to sum over multiple numbers) and vertex labels (for larger number systems than binary), e.g. in line with the experimental contributions of Alon and Yahav [2021].

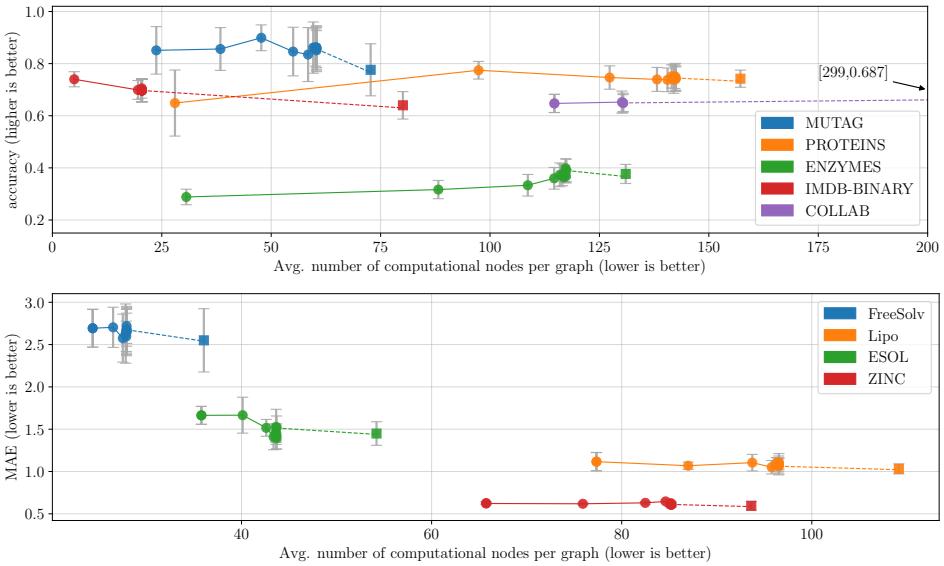

Figure 15: Computational nodes vs. accuracy. Solid segments connect models of Caterpillar GNN, with height $h = 0$ (C_0, circle) up to $h = 10$ (C_{10}, circle), the final dashed segment connects to MPGNN (square).

D.5 Experimental Setting on Real-World Datasets

We performed empirical experiments across multiple standard graph datasets, categorized by their evaluation metrics. Accuracy-based evaluation was used for bioinformatics and social network datasets, including the TUDataset [Morris et al., 2020] benchmarks such as ENZYMES, MUTAG, PROTEINS, COLLAB, and IMDB-BINARY (see Table 6). We report running times for both preprocessing and training (see Table 7). Training code, along with exact hyperparameter configurations for reproducibility, is available in the supplementary material.

For each dataset, we trained graph convolutional networks (GCNs) with variants of our proposed model Caterpillar GCN. To compare the number of saved nodes consistently, for all experiments, we *fixed the number of layers to five* ($L = 5$), gradient clipping to 1.0, and based on the validation set performance, we employed early stopping with a fixed `patience` parameter of 20 to prevent overfitting. We conducted extensive k-fold cross-validation (typically 10 folds), ensuring robust performance evaluation. For the ZINC dataset, we always used 5 random seed initializations for the public splits. Models were trained with standard settings, using the Adam optimizer [Kingma and Ba, 2017], a moderate learning rate, and weight decay to balance training stability and convergence speed. Complete hyperparameter details are provided in the supplementary material. Experiments were repeated using fixed random seeds to ensure reproducibility. All models were trained using an Intel Xeon E5-2690 @ 2.90 GHz processor (16 cores, 32 threads, 20 MB L3 cache, 64 B cache line) equipped with 64 GB of RAM.

type	MUTAG			PROTEINS			ENZYMES			IMDB-BINARY			COLLAB		
	mean ± std	#n.	%s.	mean ± std	#n.	%s.	mean ± std	#n.	%s.	mean ± std	#n.	%s.	mean ± std	#n.	%s.
$\mathcal{C}_0$	0.851 ± 0.091	23.7	67.4	0.649 ± 0.127	28.0	82.2	0.288 ± 0.030	30.6	76.6	**0.740 ± 0.029**	5.0	93.8	0.647 ± 0.035	114.7	61.6
$\mathcal{C}_1$	0.856 ± 0.082	38.4	47.2	**0.774 ± 0.034**	97.4	38.1	0.317 ± 0.035	88.2	32.7	0.699 ± 0.036	19.6	75.6	0.647 ± 0.035	114.7	61.6
$\mathcal{C}_2$	**0.899 ± 0.050**	47.7	34.4	0.747 ± 0.045	127.3	19.0	0.333 ± 0.042	108.6	17.1	0.705 ± 0.037	20.4	74.5	0.652 ± 0.042	130.1	56.5
$\mathcal{C}_3$	0.846 ± 0.093	55.0	24.4	0.739 ± 0.046	138.2	12.1	0.360 ± 0.042	114.7	12.5	0.697 ± 0.043	20.5	74.5	0.649 ± 0.032	130.4	56.4
$\mathcal{C}_4$	0.835 ± 0.103	58.4	19.7	0.738 ± 0.045	140.6	10.6	0.373 ± 0.045	116.0	11.5	0.697 ± 0.043	20.5	74.5	0.649 ± 0.037	130.4	56.4
$\mathcal{C}_5$	0.861 ± 0.098	59.6	18.1	0.751 ± 0.045	141.5	10.0	0.373 ± 0.042	116.5	11.0	0.697 ± 0.043	20.5	74.5	0.651 ± 0.035	130.4	56.4
$\mathcal{C}_6$	0.862 ± 0.071	60.1	17.4	0.739 ± 0.052	142.0	9.7	0.363 ± 0.015	116.9	10.8	0.697 ± 0.043	20.5	74.5	0.651 ± 0.035	130.4	56.4
$\mathcal{C}_7$	0.862 ± 0.083	60.3	17.1	0.751 ± 0.047	142.2	9.5	0.377 ± 0.033	117.1	10.6	0.697 ± 0.043	20.5	74.5	0.651 ± 0.035	130.4	56.4
$\mathcal{C}_8$	0.851 ± 0.076	60.3	17.0	0.748 ± 0.052	142.3	9.5	**0.398 ± 0.037**	117.2	10.5	0.697 ± 0.043	20.5	74.5	0.651 ± 0.035	130.4	56.4
$\mathcal{C}_9$	0.862 ± 0.076	60.4	17.0	0.740 ± 0.047	142.4	9.4	0.368 ± 0.026	117.4	10.4	0.697 ± 0.043	20.5	74.5	0.651 ± 0.035	130.4	56.4
$\mathcal{C}_{10}$	0.856 ± 0.087	60.4	17.0	0.745 ± 0.047	142.5	9.4	0.390 ± 0.043	117.4	10.4	0.697 ± 0.043	20.5	74.5	0.651 ± 0.035	130.4	56.4
MP	0.776 ± 0.100	72.7	0.0	0.742 ± 0.033	157.2	0.0	0.377 ± 0.037	131.0	0.0	0.640 ± 0.053	80.1	0.0	**0.687 ± 0.043**	299.0	0.0

Table 6: Results for *graph-level classification* datasets. By C_h is denoted the (caterpillar height) parameter h in $\mathbb{N}$ of graph-level aggregation (ours), while MP denotes the full message-passing GCN. We report mean validation accuracy with standard deviation across 10 different splits. Columns "#n." denote the number of *nodes* of the computation graph, and columns "%s." the percentage of nodes of the computation graph *saved* compared with message-passing (MP).

	MUTAG		PROTEINS		ENZYMES		IMDB-BINARY		COLLAB	
	features: 7	graphs: 188	features: 3	graphs: 1113	features: 3	graphs: 600	features: 0	graphs: 1000	features: 0	graphs: 5000
	avg. $\lvert V\rvert$: 17.9	avg. $\lvert E\rvert$: 39.6	avg. $\lvert V\rvert$: 39.1	avg. $\lvert E\rvert$: 145.6	avg. $\lvert V\rvert$: 32.6	avg. $\lvert E\rvert$: 124.3	avg. $\lvert V\rvert$: 19.8	avg. $\lvert E\rvert$: 193.1	avg. $\lvert V\rvert$: 74.5	avg. $\lvert E\rvert$: 4914.4
	parameters: 15.0k		parameters: 53.4k		parameters: 14.4k		parameters: 49.6k		parameters: 66.3k	
type	precomp.	avg. training	precomp.	avg. training	precomp.	avg. training	precomp.	avg. training	precomp.	avg. training
C_0	2.98 s	42.53 s	13.21 s	2.90 min	5.99 s	1.17 min	7.64 s	1.80 min	1.82 min	11.40 min
C_1	6.87 s	38.95 s	1.52 min	3.15 min	48.33 s	55.14 s	1.21 min	1.81 min	100.51 min	11.70 min
C_2	10.83 s	43.59 s	2.93 min	2.25 min	1.36 min	59.12 s	1.64 min	2.04 min	155.25 min	12.08 min
C_3	15.19 s	41.88 s	3.91 min	2.27 min	1.75 min	1.07 min	1.66 min	2.07 min	198.42 min	14.59 min
C_4	18.85 s	40.59 s	4.40 min	2.57 min	1.96 min	1.07 min	1.81 min	2.07 min	220.47 min	11.88 min
C_5	20.82 s	42.31 s	4.75 min	2.48 min	2.09 min	1.05 min	1.66 min	2.11 min	220.70 min	13.01 min
C_6	22.23 s	41.46 s	4.96 min	2.44 min	2.14 min	1.18 min	1.65 min	2.06 min	220.22 min	12.54 min
MP	—	35.96 s	—	2.49 min	—	1.08 min	—	1.59 min	—	27.78 min

Table 7: *Running times across datasets.* By C_h is denoted the (caterpillar height) parameter h in $\mathbb{N}$ of graph-level aggregation (ours), while MP denotes the full message-passing GCN. Columns "precomp" report the preprocessing time for the computation of graph-level aggregation matrices, and columns "avg. training" report the average training time across multiple runs.

type	FreeSolv			Lipo			ESOL			ZINC		
	mean ± std	#n.	%s.	mean ± std	#n.	%s.	mean ± std	#n.	%s.	mean ± std	#n.	%s.
$\mathcal{C}_0$	2.693 ± 0.223	24.3	32.5	1.116 ± 0.108	77.3	29.1	1.664 ± 0.106	35.8	34.0	0.570 ± 0.012	65.7	29.8
$\mathcal{C}_1$	2.693 ± 0.223	24.3	32.5	1.116 ± 0.108	77.3	29.1	1.664 ± 0.106	35.8	34.0	0.571 ± 0.006	65.7	29.8
$\mathcal{C}_2$	2.704 ± 0.237	26.5	26.5	1.069 ± 0.043	87.0	20.3	1.666 ± 0.212	40.1	26.0	0.584 ± 0.007	75.9	18.9
$\mathcal{C}_3$	2.576 ± 0.283	27.5	23.7	1.105 ± 0.098	93.7	14.1	1.516 ± 0.100	42.6	21.4	0.602 ± 0.003	82.5	11.9
$\mathcal{C}_4$	2.630 ± 0.350	27.8	22.8	1.051 ± 0.079	95.8	12.3	1.410 ± 0.152	43.4	20.0	0.570 ± 0.012	84.6	9.6
$\mathcal{C}_5$	2.597 ± 0.178	27.9	22.6	1.097 ± 0.067	96.3	11.8	1.458 ± 0.102	43.6	19.6	0.551 ± 0.012	85.1	9.1
$\mathcal{C}_6$	2.650 ± 0.266	27.9	22.6	1.070 ± 0.077	96.5	11.6	1.528 ± 0.207	43.6	19.5	0.546 ± 0.013	85.2	9.0
$\mathcal{C}_7$	2.719 ± 0.206	27.9	22.6	1.072 ± 0.111	96.5	11.6	1.446 ± 0.101	43.7	19.4	0.539 ± 0.012	85.2	9.0
$\mathcal{C}_8$	2.676 ± 0.196	27.9	22.6	1.108 ± 0.104	96.5	11.6	**1.392 ± 0.129**	43.7	19.4	0.543 ± 0.006	85.2	9.0
$\mathcal{C}_9$	2.652 ± 0.284	27.9	22.6	1.094 ± 0.069	96.5	11.6	1.416 ± 0.148	43.7	19.4	0.538 ± 0.003	85.2	9.0
$\mathcal{C}_{10}$	2.675 ± 0.276	27.9	22.6	1.063 ± 0.091	96.5	11.6	1.514 ± 0.142	43.7	19.4	0.528 ± 0.004	85.2	9.0
MP	**2.550 ± 0.374**	36.0	0.0	**1.031 ± 0.059**	109.2	0.0	1.449 ± 0.140	54.2	0.0	**0.477 ± 0.020**	93.6	0.0

Table 8: Results for *graph-level regression* datasets. By C_h is denoted the (caterpillar height) parameter h in $\mathbb{N}$ of graph-level aggregation (ours), while MP denotes the full message-passing GCN. We report validation mean absolute error (MAE) with standard deviation across 10 different splits and random seeds, with five repetitions using different random seeds for ZINC. Columns "#n." denote the number of *nodes* of the computation graph, and columns "%s." the percentage of nodes of the computation graph *saved* compared with message-passing (MP).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions: graph-level aggregation, the Caterpillar GNN architecture, the expressivity characterization via generalized caterpillar graphs, and the empirical study of accuracy–efficiency trade-offs. These claims are supported by the theoretical results in Sections 3, 4 and the experiments in Section 5 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion states that the current expressivity characterization is restricted to undirected graphs and does not directly extend to directed graphs. The experimental section also presents the results as dataset-dependent trade-offs rather than universal improvements.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main theoretical results state their assumptions in the theorem statements and are accompanied by proof sketches in the main text where appropriate. Complete proofs are provided in Appendices A and B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the model construction, datasets, evaluation setting, and training procedure, with further details in Appendix D. The supplementary material provides training code and exact hyperparameter configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in Appendix D.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix D specifies the experimental setup, including datasets, cross-validation or public splits, fixed number of layers, optimizer, early stopping, random seeds, and hyperparameter details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experimental tables report mean performance with standard deviation across different splits and random seeds, or across repeated random seed initializations for datasets with public splits. These statistics are reported for the experiments supporting the main empirical claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D reports the compute resources used for the experiments, including hardware specifications. The appendix also reports preprocessing and training times for the experimental runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is foundational research on graph neural network architectures and expressivity. It uses standard public datasets and synthetic benchmarks, and we are not aware of any aspect of the work that violates the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The work is foundational and methodological, studying graph neural network expressivity and computational graphs rather than a deployed system or application-specific model. We do not identify a direct path to specific societal harms beyond the general downstream uses of graph learning methods.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models, pre-trained generative models, or scraped datasets. The released material concerns code for a graph-learning method and experiments on standard public or synthetic datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the original sources for the public datasets and software packages used in the experiments. The datasets are standard graph-learning benchmarks, and the paper does not repackage or redistribute them as new assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper introduces code for the proposed method and a synthetic benchmark generation procedure. These are documented in the supplementary material together with implementation and experimental details.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development, theoretical results, and experiments do not involve LLMs as an important, original, or non-standard component. Any use of LLMs, if applicable, was limited to writing, editing, or formatting assistance and did not affect the methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.