Definable decompositions for graphs of bounded linear cliquewidth[*]

Mikołaj Bojańczyk
Institute of Informatics, University of Warsaw, Poland
bojan@mimuw.edu.pl

Martin Grohe
Department of Computer Science, RWTH Aachen University, Germany
grohe@informatik.rwth-aachen.de

Michał Pilipczuk
Institute of Informatics, University of Warsaw, Poland
michal.pilipczuk@mimuw.edu.pl

Abstract

We prove that for every positive integer k, there exists an $\textsc{mso}_1$-transduction that given a graph of linear cliquewidth at most k outputs, nondeterministically, some clique decomposition of the graph of width bounded by a function of k. A direct corollary of this result is the equivalence of the notions of $\textsc{cmso}_1$-definability and recognizability on graphs of bounded linear cliquewidth.

1 Introduction

Hierarchical decompositions of graphs have come to play an increasingly important role in logic, algorithms and many other areas of computer science. The treelike structure they impose often allows to process the data much more efficiently. The best-known and arguably most important graph decompositions are *tree decompositions*, which play a central role in a research direction at the boundary between logic, graph grammars, and generalizations of automata theory to graphs that was pioneered by Courcelle in the 1990s (see [6]).

A drawback of tree decompositions is that they only yield meaningful results for sparse graphs. A suitable form of decomposition that also applies to dense graphs and that has a similarly nice, yet less developed Courcelle-style theory, is that of *clique decompositions*, introduced by Courcelle and Olariu [8]. A clique decomposition of a graph is a term in a suitable algebra consisting of (roughly) the following operations for constructing and manipulating colored graphs: (i) disjoint union; (ii) for a pair of colors i, j, simultaneously add an edge for every pair (i-colored vertex, j-colored vertex); and (iii) apply recolorings to entire colors. A natural notion of *width* for such a clique decomposition is the total number of colors used. The *cliquewidth* of a graph is the smallest width of a clique decomposition for it. An alternative notion of graph decomposition that also works well for dense graphs is that of *rank decompositions* introduced by Oum and Seymour [12, 13]. The corresponding notion of *rankwidth* turned out to be functionally equivalent to cliquewidth, that is, bounded cliquewidth is the same as bounded rankwidth. Bounded width clique or rank decompositions may be viewed as hierarchical decompositions that minimize "modular complexity" of cuts in the decomposition, in the same way as treewidth corresponds to hierarchical decompositions using vertex cuts, where the complexity of a cut is its size.

Cliquewidth is tightly connected to $\textsc{mso}_1$ logic on graphs, in the same way as the $\textsc{mso}_2$ logic is connected to treewidth. Recall that in $\textsc{mso}_1$, one can quantify over vertices and sets of vertices, and check their adjacency, while $\textsc{mso}_2$ also allows quantification over sets of edges. Both these logics can be viewed as plain $\textsc{mso}$ logic on two different encodings of graphs as relational structures: for $\textsc{mso}_1$ the encoding uses only vertices as the universe and has a binary adjacency relation, while for $\textsc{mso}_2$ the encoding uses both vertices and edges as the universe and has an incidence relation binding every edge with its endpoints. These two logics are connected to cliquewidth and treewidth as follows. If a graph property Π is definable in $\textsc{mso}_2$, then tree decompositions of graphs in Π can be recognized by a finite state device (tree automaton). This leads, for instance, to a fixed-parameter model checking algorithm for $\textsc{mso}_2$-definable properties on graphs of bounded treewidth [5]. This notion of recognizability, where tree decompositions are processed, is called *HR-recognizability* [6]. Similarly, if Π is $\textsc{mso}_1$-definable, then clique decompositions of graphs in Π can be recognized by a finite state device. This notion of recognizability is called *VR-recognizability* [6], and it yields a fixed-parameter model checking algorithm for $\textsc{mso}_1$-definable properties on graphs of bounded cliquewidth [7].

Courcelle [5] conjectured that $\textsc{mso}_2$-definability and recognizability for tree decompositions (i.e. HR-recognizability) are equivalent for every graph class of bounded treewidth, provided that $\textsc{mso}_2$ is extended by counting predicates of the form "the size of X is divisible by p", for every integer p (this logic is called $\textsc{cmso}_2$). This conjecture has been resolved by two of the current authors [3]. More precisely, in [3] it was shown that for every k there exists an $\textsc{mso}$ transduction which inputs a graph of treewidth at most k and (nondeterministically) outputs its tree decomposition of width bounded by a function of k. The graph is given via its incidence encoding. The conjecture of Courcelle then follows by composing this transduction with guessing the run of an automaton recognizing the property in question on the output decomposition.

The same question can be asked about cliquewidth: is it true that every property of graphs of bounded cliquewidth is $\textsc{mso}_1$-definable if and only if it is (VR-)recognizable? The present paper discusses this question, proving a special case of the equivalence, namely for bounded linear cliquewidth.

Our contribution. Our main result (Theorem 3.3) is that for every $k \in \mathbb{N}$, there is an $\textsc{mso}$-transduction that inputs a graph of *linear* cliquewidth at most k, and outputs a clique decomposition of it

[*]The work of M. Bojańczyk and Mi. Pilipczuk was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC consolidator grant LIPA, agreement no. 683080). Mi. Pilipczuk was supported by the Foundation for Polish Science via the START stipend programme.

LICS '18, July 9–12, 2018, Oxford, United Kingdom
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209135

which has width bounded by a function of k. Here, we use the adjacency encoding of the graph. The *linear cliquewidth* of a graph is a linearized variant of cliquewidth, similarly as pathwidth is a linearized variant of treewidth; see Section 2 for definition and [1, 9–11] for more background. A direct consequence of this result (Theorem 3.5) is that every property of graphs of bounded linear cliquewidth is CMSO$_1$-definable if and only if it is (VR-)recognizable. This gives a partial positive answer to the question above.

The proof of our main result shares one key idea with the proof of the MSO$_2$-definability of tree decompositions, or more precisely, its pathwidth part [3, Lemma 2.5]. This is the use of Simon's Factorization Forest Theorem [14]. We view a linear clique decomposition of width k as a word over a finite alphabet and use the factorization theorem to construct a nested factorization of this word of depth bounded in terms of k. The overall MSO transduction computing a decomposition is then constructed by induction on the nesting depth of this factorization. The technical challenge in this paper is to analyze the composition of "subdecompositions", which is significantly more complicated in the cliquewidth case than in the treewidth/pathwidth case of [3]. In a path decomposition, each node of the path (over which we decompose) naturally corresponds to a separation of the graph, with the bag at the node being the separator. Thus, in the pathwidth case, each separation appearing in the decomposition essentially can be described by a tuple of vertices in the separator, with the left and the right side being essentially independent; this is a simple and easy to handle object. The difficulty in the cliquewidth case is that "separations" appearing in a linear clique decomposition are partitions of the vertex set into two sides with small "modular complexity": each side can be partitioned further into a bounded number of parts so that vertices from the same part have exactly the same neighbors on the other side. Such separations are much harder to control combinatorially, and hence capturing them using the resources of MSO requires more insight into the combinatorics of linear cliquewidth.

Full version. In this extended abstract we only sketch our results, while complete arguments can be found in the full version, which is available as a preprint on arXiv [2]. Statements whose proofs are deferred to the full version are marked with $\star$.

2 Preliminaries

Graphs and cliquewidth. All graphs considered in this paper are finite and simple. For the most part we use undirected graphs, if we use directed graphs then we remark this explicitly. We write $[k]$ for $\{1, 2, \ldots, k\}$ and $\binom{X}{1,2}$ for the family of nonempty subsets of a set X of size at most 2.

A *k-colored graph* is a graph with each vertex assigned a color from $[k]$. On k-colored graphs we define the following operations.

- **Recolor.** For every function $\phi\colon [k] \to [k]$ there is a unary operation which inputs one k-colored graph and outputs the same graph where each vertex is recolored to the image of its original color under ϕ.
- **Join.** For every family of subsets $S \subseteq \binom{[k]}{1,2}$ there is an operation that inputs a family of k-colored graphs, of arbitrary finite size, and outputs a single k-colored graph constructed as follows. Take the disjoint union of the input graphs and for each $\{i, j\} \in S$ (possibly $i = j$), add an edge between every pair of vertices that have colors i and j, respectively, and originate from different input graphs.
- **Constant.** For each color $i \in [k]$ there is a constant which represents a graph on a single vertex with color i.

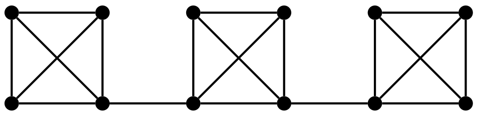

Figure 1. A graph of linear cliquewidth 3

Define a *width-k clique decomposition* to be a (rooted) tree where nodes are labelled by operation names in an arity preserving way, that is, all constants are leaves and all recolor operations have exactly one child. The tree does not have any order on siblings, because Join is a commutative operation. For a clique decomposition, we define its *result* to be the k-colored graph obtained by evaluating the operations in the decomposition. The *cliquewidth* of a graph is the minimum number k for which there is a width-k clique decomposition whose result is (some coloring of) the graph.

We remark that we somewhat diverge from the original definition of cliquewidth [8] in the following way. In [8], there is one binary disjoint union operation that just adds two input k-colored graphs, and for each pair of different colors i, j there is a unary operation that creates an edge between every pair of vertices of colors i and j, respectively. For our purposes, we need to have a union operation that takes an arbitrary number of input graphs. This is because an MSO transduction constructing a clique decomposition cannot break symmetries and join isomorphic parts of the graph in some arbitrarily chosen order, which would be necessary if we used binary disjoint union operations. Another difference is that our join does simultaneously two operations: it takes the disjoint union of several inputs, and adds edges between them. When using binary joins, the two operations can be separated by introducing temporary colors; however when the number of arguments is unbounded such a separation is not possible. It is easy to show that our definition of cliquewidth is at multiplicative factor at most 2 from the original definition.

Linear cliquewidth is a linearized variant of cliquewidth, where we allow only restricted joins that add only a single vertex. More precisely, we replace the Join and Constant operations with one unary operation **Add Vertex**. This operation is parameterized by a color $i \in [k]$ and a color subset $X \subseteq [k]$, and it adds to the graph a new vertex of color i, adjacent exactly to vertices with colors belonging to X. A *width-k linear clique decomposition* is a word consisting of Add Vertex and Recolor operations, and the *result* of such a decomposition is the k-colored graph obtained by evaluating the operations over the empty graph. The *linear cliquewidth* of a graph is defined just like cliquewidth, but we consider only linear clique decompositions. Note that we can transform any linear clique decomposition of width k to a clique decomposition of width at most $(k + 1)$ by replacing each subterm $\textbf{Add Vertex}_{i,X}(\theta)$ by the term

$$\textbf{Recolor}_{j\mapsto i}\Big(\textbf{Join}_{\{\{j,x\}\colon x\in X\}}\big(\theta, \textbf{Constant}_j\big)\Big),$$

where j is a color not occurring in θ. It follows that the linear cliquewidth of a graph is at least its cliquewidth minus one.

Example 2.1. Consider the graph G displayed in Figure 1. We first argue that its cliquewidth is at most 3, and then show that also its linear cliquewidth is at most 3.

Let us construct a clique decomposition of G of width 3. The following three terms $\theta_1, \theta_2, \theta_3$ construct the three 4-cliques of G with appropriate colors:

$$\theta_1 = \mathbf{Join}_{\{\{1\},\{1,2\}\}}\Big(\Big\{\mathbf{Color}_1, \mathbf{Color}_1, \mathbf{Color}_1, \mathbf{Color}_2\Big\}\Big),$$

$$\theta_2 = \mathbf{Join}_{\{\{1\},\{1,2\},\{1,3\},\{2,3\}\}}\Big(\Big\{\mathbf{Color}_1, \mathbf{Color}_1, \mathbf{Color}_2, \mathbf{Color}_3\Big\}\Big),$$

$$\theta_3 = \mathbf{Join}_{\{\{1\},\{1,3\}\}}\Big(\Big\{\mathbf{Color}_1, \mathbf{Color}_1, \mathbf{Color}_1, \mathbf{Color}_3\Big\}\Big).$$

Then the term

$$\mathbf{Join}_{\{\{2\},\{3\}\}}\left(\{\theta_1, \theta_2, \theta_3\}\right)$$

is a width-3 clique decomposition of the graph G.

For a linear clique decomposition, it is convenient to denote the unary operation $\mathbf{Add\ Vertex}_{i,X}$ of adding a vertex of color i and connecting it to all vertices of color X by $\boldsymbol{a}_{i,X}$ and the recoloring operation by $\boldsymbol{r}_\phi$. Moreover, we apply both operations by multiplication from left to right, omitting parenthesis. This way, we may view a linear clique decomposition of width k as a word over the finite alphabet

$$\{\boldsymbol{a}_{i,X} \mid i \in [k], X \subseteq 2^{[k]}\} \cup \{\boldsymbol{r}_\phi \mid \phi\colon [k] \to [k]\}.$$

With this notation, our linear clique decomposition of the graph G is the concatenation of the following three sequences of operations, constructing the three consecutive cliques:

$$\begin{aligned} \text{first clique:}\quad & \boldsymbol{a}_{1,\emptyset}\boldsymbol{a}_{1,\{1\}}\boldsymbol{a}_{1,\{1\}}\boldsymbol{a}_{2,\{1\}}, \\ \text{second clique:}\quad & \boldsymbol{a}_{3,\{2\}}\boldsymbol{r}_{\{2\mapsto 1, 3\mapsto 2\}}\boldsymbol{a}_{2,\{2\}}\boldsymbol{a}_{2,\{2\}}\boldsymbol{a}_{3,\{2\}}\boldsymbol{r}_{\{2\mapsto 1, 3\mapsto 2\}}, \\ \text{third clique:}\quad & \boldsymbol{a}_{3,\{2\}}\boldsymbol{a}_{3,\{3\}}\boldsymbol{a}_{3,\{3\}}\boldsymbol{a}_{3,\{3\}}. \end{aligned}$$

Relational structures and logic. A *vocabulary* is a set of *relation names*, each one with associated arity that is a nonnegative integer. A *relational structure* over the vocabulary Σ consists of a set called the *universe*, and for each relation name in the vocabulary, an associated relation of the same arity over the universe. Note the possibility of relations of arity zero, such a relation stores a single bit of information about the structure. A graph is encoded as a relational structure as follows: the universe is the vertex set and there is one symmetric binary relation encoding adjacency.

A width-k clique decomposition of a graph is modeled as a relational structure whose universe is the set of nodes of the decomposition, there is a binary predicate "child", and for each operation from the definition of a clique decomposition there is a unary predicate (the set of these predicates depends on k) which selects nodes that use this operation. Note that the graph itself is not included in this structure, but it is straightforward to reconstruct it using an MSO transduction (see below).

To describe properties of relational structures, we use monadic second-order logic (MSO). This logic allows quantification both over single elements of the universe and also over subsets of the universe. For a precise definition of MSO, see [6]. We also use counting MSO, denoted also by CMSO, which is the extension of MSO with predicates of the form "the size of X is divisible by p" for every $p \in \mathbb{N}$.

MSO transductions. We use the same notion of MSO transductions as in [3, 4]. A transduction with input vocabulary Σ and output vocabulary Γ is a set of pairs

(input structure over Σ, output structure over Γ)

which is invariant under isomorphism of relational structures. Note that a transduction is a relation and not necessarily a function, thus it can have many different (or zero) possible outputs for the same input. An MSO transduction is any transduction that can be obtained by composing a finite number of *atomic transductions* of the following kinds (precise definitions can be found in the full version):

1. **Filtering.** Check if some MSO sentence is satisfied in the structure and discard it if it is not the case.
2. **Restriction.** Restrict the universe to elements satisfying some MSO formula with one free variable
3. **Mso interpretation.** Add a new relation to the structure with interpretation given by an MSO formula.
4. **Copying.** Copy the structure a fixed number of times.
5. **Coloring.** Nondeterministically add a unary predicate to the structure; for every possible such unary predicate there is one output.

Note that kind 1 is a partial function, kinds 2, 3, 4 are functions, and kind 5 is a relation. While our MSO transductions differ syntactically from those used in the literature, see e.g. Courcelle and Engelfriet [6], they describe the same class of transductions.

An MSO-transduction is *deterministic* if it uses no coloring. A deterministic MSO-transduction is a partial function, that is, for each input structure there is at most one output structure.

Each element v' of an output of an MSO-transduction is either equal to some element v of the input structure, or is a copy of some element v of the input structure. We call this element v the *origin* of v'. Thus we have a well-defined *origin mapping* from the output structure to the input structure. In general, this mapping is neither injective nor surjective.

Define the *size* of an atomic MSO transduction to be the size of its input and output vocabularies, plus the maximal quantifier rank of MSO formulas that appear in it (if the atomic type uses MSO formulas). Let the *size* of an MSO transduction to be the sum of sizes of atomic transductions that compose to the transduction. Note that there are finitely many MSO transductions of a given size, as there are finitely many MSO formulas (up to logical equivalence) once the vocabulary, the free variables, and the quantifier rank are fixed.

The composition of two MSO transductions is an MSO transduction by definition. Another property that we use, as expressed in the following lemma, is that the union of two MSO transductions is also an MSO transduction; recall that here we regard MSO transductions as relations between input and output structures. This property is Lemma 7.18 from [6]. Since our notion of an MSO transduction is a bit different from the one used in [6], we give a proof in the full version.

Lemma 2.2 (⋆). *The union of two* MSO *transductions with same input and output vocabularies is also an* MSO *transduction.*

The key property of MSO transductions is that CMSO- and MSO-definable properties are closed under taking inverse images over MSO transductions. More precisely, we have:

Lemma 2.3 (Backwards Translation Theorem, [6]). *Let $\mathcal{I}$ be an* MSO *transduction with input vocabulary Σ and output vocabulary Γ. Then for every* MSO *(resp.* CMSO*) sentence ψ over Γ there exists an*

MSO *(resp.* CMSO*) sentence $\varphi = \mathcal{I}^{-1}(\psi)$ over Σ such that φ holds in exactly those Σ-structures on which $\mathcal{I}$ produces at least one output satisfying ψ.*

Simon Lemma. As we mentioned in Section 1, the main technical tool used in this work will be Simon's Factorization Theorem [14]. We use the following variant, which is an easy corollary of the original statement. Recall that a *semigroup* is an algebra with one associative binary operation, usually denoted as multiplication, and that an *idempotent* in a semigroup is an element e such that $e \cdot e = e$.

Lemma 2.4 (Simon Lemma, ⋆)**.** *Suppose that S and T are semigroups, where S is finitely generated (but possibly infinite) and T is finite. Suppose further that $h\colon S \to T$ is a semigroup homomorphism and $f\colon \mathbb{N} \to \mathbb{N}$ and $\mu\colon S \to \mathbb{N}$ are functions such that*

$$\mu(s_1 \cdot \ldots \cdot s_n) \leqslant f(\max_{i \in [n]} \mu(s_i)) \tag{1}$$

holds whenever $n = 2$ or there is some idempotent $e \in T$ such that $e = h(s_1) = \ldots = h(s_n)$. Then μ has finite range, i.e. there exists $K \in \mathbb{N}$ such that $\mu(s) \leqslant K$ for all $s \in S$.

3 Main results

Statement of the main result. Our main result is that for every k, there is an MSO transduction which maps every graph of linear cliquewidth k to a nonempty set of its clique decompositions. The width of these decompositions is bounded by a function of k; we do not achieve the optimal value k. To state this result, we introduce a graph parameter, called *definable cliquewidth*, which measures the size of an MSO transduction necessary to transform the graph into its clique decomposition.

Recall that we model a width-k clique decomposition of a graph as a (rooted) tree labelled by an alphabet of operations depending on k. Such a clique decomposition t constructs a graph G_t whose vertices are the leaves of the tree. More generally, we say that t is a clique decomposition of a graph G if there is an isomorphism from G_t to G.

Definition 3.1 (Decomposer)**.** A *width-k decomposer* is an MSO transduction $\mathcal{D}$ from the vocabulary of graphs to the vocabulary of width-k clique decompositions such that for every input-output pair (G, t) of $\mathcal{D}$ the following two conditions are satisfied.

(a) t is a width-k clique decomposition of G.
(b) The origin mapping from t to G restricted to the leaves of t is an isomorphism from G_t to G.

Condition (b) in the definition of decomposers may seem unnecessarily restrictive, but it will turn out to be very useful in the technical arguments. Furthermore, natural transductions satisfying (a) also tend to satisfy (b), because usually such transduction proceed by building the tree of a clique decomposition on top of the input graph.

Definition 3.2. The *definable cliquewidth* of a graph G, denoted by $\mathrm{dcw}(G)$, is the smallest size of a decomposer which produces at least one output on G.

Observe that the size of a decomposer (as a particular MSO-transduction) is an upper bound for its width, as the size of a transduction is larger than the size of its output vocabulary. Thus, the definable cliquewidth of a graph is always at least its cliquewidth.

Note that there are finitely many decomposers of a given size, and decomposers are closed under union by Lemma 2.2. Therefore, for every k there is a single decomposer which produces at least one output on every graph with definable cliquewidth at most k, namely one can take the union of all decomposers of size at most k.

The main technical result of this paper is the following.

Theorem 3.3. *For every $k \in \mathbb{N}$ there exists a decomposer $\mathcal{D}$ which has at least one output for every graph of linear cliquewidth at most k. In other words, the definable cliquewidth of a graph is bounded by a function of its linear cliquewidth.*

This result could be improved in two ways: first, we could make the transduction produce results for graphs of bounded cliquewidth (and not bounded linear cliquewidth), and second, we could produce clique decompositions of optimum width. We leave both of these improvements to future work. Note that it is impossible to find a decomposer which produces a linear clique decomposition for every graph of linear cliquewidth k; the reason is that such a decomposer would impose a total order on the vertex set, and this is impossible for some graphs, e.g. large independent sets.

We remark that, similarly to the case of treewidth [3], our proof is effective: the decomposer $\mathcal{D}$ can be computed from k. In this extended abstract we omit the discussion of computability issues and refer the reader to the full version for details.

Recognizability. We now state an important corollary of the main theorem, namely that for graph classes with linear cliquewidth, being definable in (counting) MSO is the same as being recognizable. Let us first define the notion of recognizability that we use. For $k \in \mathbb{N}$, define a *k-context* to be a width-k clique decomposition with one distinguished leaf. If t is a k-context and G is a k-colored graph, then $t[G]$ is defined to be the k-colored graph obtained by replacing the distinguished leaf of t by G, and then applying all the operations in t.

Definition 3.4 (Recognizability, see [6], Def. 4.29)**.** Let L be a class of graphs. Two k-colored graphs G_1,G_2 are called L-equivalent if for every k-context t we have $t[G_1] \in L$ iff $t[G_2] \in L$, where membership in L is tested after ignoring the coloring. We say that L is *recognizable* if for every $k \in \mathbb{N}$ there are finitely many equivalence classes of L-equivalence.

Theorem 5.68(2) in [6] shows that if a class of graphs is definable in MSO (in the sense used here, i.e., MSO_1), then it is recognizable (in the sense of Definition 3.4). The converse implication is not true, e.g., there are uncountably many recognizable graph classes. It is a simple corollary of our main theorem that the converse implication is true under the assumption of bounded linear cliquewidth.

Theorem 3.5. *If L is a class of graphs of bounded linear cliquewidth, then L is recognizable if and only if it is definable in* CMSO*.*

Proof. As mentioned above, the right-to-left implication is true even without assuming a bound on linear cliquewidth. For the converse, we use the following claim; since the proof is completely standard, we only sketch it.

Claim 1. *If a class of graphs L is recognizable, then for every k the following language L_k of labelled trees is definable in* CMSO*.*

$$L_k = \quad \{t\colon t \text{ is a tree that is a width-}k \text{ clique decomposition whose resulting graph is in } L\}$$

Proof sketch. The language L_k is a set of (unranked) trees without sibling order. Define $\tilde{L}_k$ to be the language of sibling-ordered trees such that if the sibling order is ignored, then the resulting tree belongs L_k. Using the assumption that L is recognizable, one shows that $\tilde{L}_k$ is definable in MSO; the idea is that using the sibling order an MSO formula can convert a tree into one which has binary branching, and then compute for each subtree its L-equivalence class. As shown in [5], if a language of sibling-ordered trees is definable in MSO and invariant under reordering siblings, then the language of sibling-unordered trees obtained from it by ignoring the sibling order is definable in CMSO without using the sibling order. Applying this to $\tilde{L}_k$ and L_k we obtain the claim. ⌟

Using Claim 1, we complete the left-to-right implication. Assume every graph from L has linear cliquewidth at most k. Apply Theorem 3.3, yielding a decomposer $\mathcal{D}$ from graphs to width-ℓ clique decompositions which produces at least one output for every graph in L. Apply Claim 1 to L and ℓ. Since $\mathcal{D}$ produces at least one output for every graph in L, we have that L is the inverse image under $\mathcal{D}$ of the language L_ℓ in the conclusion of the claim. It follows from the Backwards Translation Theorem that L is definable in CMSO. □

4 The proof strategy

In this section we present the proof strategy for our main contribution, Theorem 3.3.

A linear clique decomposition of width k, being a single path, can be viewed as a sequence of instructions. For such sequences of instructions (actually, for a similar but slightly more general object), we will use the name *k-derivations*. Intuitively speaking, a k-derivation corresponds to an infix of a linear clique decomposition of width k. We can concatenate k-derivations, which means that the set of k-derivations is endowed with a semigroup structure. The main idea is to use Simon's Factorization Theorem [14], in the flavor delivered by the Simon Lemma (Lemma 2.4), to factorize this product into a tree of bounded depth, so that definable clique decompositions of factors can be constructed via a bottom-up induction over the factorization.

Roughly speaking, the Simon Lemma is used to prove Theorem 3.3 as follows. As the semigroup S we use k-derivations. As the homomorphism h, we use a notion *abstraction*, which maps each k-derivation to a bounded-size combinatorial object consisting of all the information we need to remember about it. Composing k-derivations naturally corresponds to composing their abstractions, which formally means that the set of abstractions, whose size is bounded in terms of k, can be endowed with a semigroup structure so that taking an abstraction of a k-derivation is a semigroup homomorphism. By taking μ to be the definable cliquewidth of a graph, we use the Simon Lemma to show that μ has a finite range on the set of all k-derivations, i.e. there is a finite upper bound on the definable cliquewidth of all k-derivations. To this end, we need to prove that the assumptions of the Simon Lemma are satisfied, that is, condition (1) is satisfied when either $n = 2$ or all the abstractions of all k-derivations in the product are equal to some idempotent in the semigroup of abstractions.

Below we give a more detailed implementation of the plan, including definitions of k-derivations and their abstractions. For the rest of the paper we fix $k \in \mathbb{N}$. Our goal is to show that graphs of linear cliquewidth at most k have bounded definable cliquewidth.

Derivations. We first introduce k-derivations.

Definition 4.1. A k-derivation σ is a triple (G, λ, ϕ), where

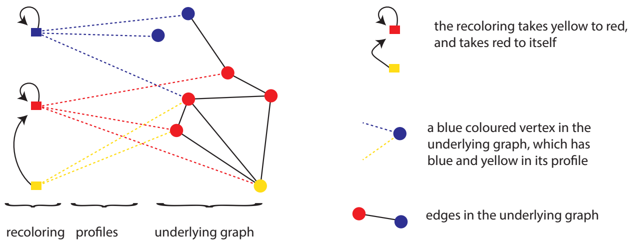
the recoloring takes yellow to red, and takes red to itself
a blue coloured vertex in the underlying graph, which has blue and yellow in its profile
edges in the underlying graph
recoloring
profiles
underlying graph

Figure 2. A k-derivation for $k = 3$, with the numbers $\{1, 2, 3\}$ being represented as colors {red, blue, yellow}. The red boxes indicate colors as used by the profiles and recoloring, and the circles indicate vertices of the underlying graph.

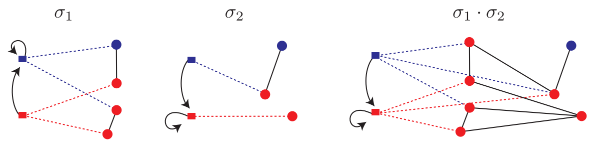

Figure 3. Composition of derivations.

- G, the *underlying graph of* σ, is a k-colored graph;
- $\lambda \colon V(G) \to 2^{[k]}$ is a function that assigns to each vertex u its *profile* $\lambda(u) \subseteq [k]$; and
- $\phi \colon [k] \to [k]$ is a function called the *recoloring*.

Intuitively, if we treat a k-derivation $\sigma = (G, \lambda, \phi)$ as a subword of instructions in a linear clique decomposition, then G is the subgraph induced by vertices introduced by these instructions and ϕ is the composition of all recolorings applied. The profile λ has the following meaning: supposing there were some instructions preceding the k-derivation in question, it assigns each vertex u of G a subset $\lambda(u)$ of colors such that among vertices introduced by these preceding instructions, u is adjacent exactly to vertices with colors from $\lambda(u)$. See Figure 2 for an example.

By the definable cliquewidth of a k-derivation we mean the definable cliquewidth of its underlying graph, with the colors ignored. For a k-derivation $\sigma = (G, \lambda, \phi)$ and $c = (i, X) \in [k] \times 2^{[k]}$, the set of all vertices with color i and profile X is called the *c-cell*, and denoted by $\sigma[c]$. For brevity, we denote $\mathscr{C}_k = [k] \times 2^{[k]}$ and interpret it as the index set of cells in k-derivations. By abuse of notation, we use the term *cell* also for the elements of $\mathscr{C}_k$.

We now describe the semigroup structure of k-derivations. We define the *composition* $\sigma_1 \cdot \sigma_2$ of two k-derivations $\sigma_1 = (G_1, \lambda_1, \phi_1)$ and $\sigma_2 = (G_2, \lambda_2, \phi_2)$ as follows; see Figure 3 for an illustration. The underlying graph of the composition is constructed by taking the disjoint union of $\phi_2(G_1)$ and G_2, where $\phi_2(G_1)$ denotes G_1 with the color of each vertex substituted with its image under ϕ_2, and adding an edge between a vertex $u \in G_1$ and a vertex $v \in G_2$ whenever the color of u in G_1 belongs to the profile $\lambda_2(v)$. The profile of a vertex u in the composition is equal to $\lambda_1(u)$ if u originates from G_1, and to $\phi_1^{-1}(\lambda_2(u))$ if u originates from G_2. Finally, the recoloring in the composition is the composition of recolorings, that is, $\phi_2 \circ \phi_1$. It is straightforward to see that composition is associative, and hence it turns the set of k-derivations into a semigroup.

Define an *atomic* k-derivation to be one where the underlying graph has at most one vertex. The number of different atomic k-derivations is finite and bounded only in terms of k, because the only freedom is the choice of the color and the profile of the unique vertex (if there is one), as well as the recoloring. Define S_k to be the subsemigroup of the semigroup of all k-derivations which is generated by the atomic k-derivations. By definition, S_k is finitely generated. The following claim is a straightforward reformulation of the definition of linear cliquewidth.

Lemma 4.2 (⋆). *If a graph has linear cliquewidth at most k, then it is the underlying graph of some k-derivation $\sigma \in S_k$.*

Abstractions. Our goal is to apply the Simon Lemma to the finitely generated semigroup S_k, with μ being the definable cliquewidth of the underlying graph. To apply the Simon Lemma, we also need a homomorphism from S_k to some finite semigroup. This homomorphism is going to be abstraction, and we define it below.

To define the abstraction, we need one more auxiliary concept, namely the *flipping* operation on a graph. For a graph G and vertex subsets $X, Y \subseteq V(G)$, the *flip between X and Y* is defined to be the following operation modifying G: for each $x \in X$, $y \in Y$, $x \neq y$, if there is an edge xy then remove it, and otherwise add it. In other words, flipping between X and Y means reversing the adjacency relation in all pairs of different elements from $X \times Y$. Note that in the flip operation, the sets X and Y need not be disjoint. Suppose that σ is a k-derivation. Recall that $\mathscr{C}_k$ represents the names of cells, i.e. each element of $\mathscr{C}_k$ is a pair (vertex color, profile). For a subset $Z \subseteq \binom{\mathscr{C}_k}{1,2}$ define the *Z-flip of σ* to be the graph obtained from the underlying graph of G by performing the flip between $\sigma[c]$ and $\sigma[d]$ for each $\{c, d\} \in Z$. Note Z can contain singletons, i.e. we might have $c = d$.

Definition 4.3. For a k-derivation σ, its *abstraction*, denoted $[\![\sigma]\!]$, is the triple (L, ρ, ϕ) consisting of the following:

- $L \subseteq \mathscr{C}_k$ is the set of cells that are nonempty in σ, called *essential*;
- $\rho \subseteq 2^{\binom{\mathscr{C}_k}{1,2}} \times \mathscr{C}_k \times \mathscr{C}_k \times 2^{\mathscr{C}_k}$ is the *connectivity registry*, which contains all tuples (Z, c, d, W) such that the Z-flip of σ satisfies: there exits a path that starts in some vertex of $\sigma[c]$, ends in some vertex of $\sigma[d]$, and whose all internal vertices belong to $\bigcup_{b \in W} \sigma[b]$;
- ϕ is the recoloring function of σ.

We explain the idea behind the connectivity registry. In general, we would like to remember which pairs of cells can be connected by a path in the underlying graph of the derivation. However, in the proof we will sometimes work not with a k-derivation, but with some Z-flip of it. Therefore, we want the abstraction to store the connectivity information after every possible flip. For technical reasons, we also remember the subset of cells traversed by the path.

Denote by T_k the set of all possible abstractions of k-derivations; note that T_k is a finite set whose size depends only on k, albeit it is doubly exponential in k. We leave it to the reader to prove that "having the same abstraction" is a congruence in the semigroup S_k, that is, an equivalence relation $\sim$ on S_k such that $s \sim s'$ and $t \sim t'$ imply $st \sim s't'$ for all $s, s', t, t' \in S_k$. It follows that we may endow T_k with a unique binary composition operation which makes it into a semigroup, and which makes the abstraction function a semigroup homomorphism from S_k to T_k.

Applying the Simon Lemma. We will apply the Simon Lemma for $S = S_k$, $T = T_k$, h being the abstraction, and μ being the definable cliquewidth of the underlying graph of a k-derivation (after forgetting the coloring). The conclusion of the Simon Lemma will say that μ has bounded range, i.e. there is a finite bound on the definable cliquewidth of the underlying graphs of derivations from S_k. Since these underlying graphs are the same as graphs of linear cliquewidth at most k by Lemma 4.2, this will mean that bounded linear cliquewidth implies bounded definable cliquewidth, thus proving Theorem 3.3.

To apply the Simon Lemma, we need to verify that assumption (1) is satisfied for some function $f : \mathbb{N} \to \mathbb{N}$. The treatment of cases when $n = 2$, and when all derivations have a common idempotent abstraction, is different, as expressed in the following two lemmas.

Lemma 4.4 (Binary Lemma). *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that*

$$\mathrm{dcw}(\sigma \cdot \tau) \leqslant f(\max(\mathrm{dcw}(\sigma), \mathrm{dcw}(\tau)))$$

for every $\sigma, \tau \in S_k$.

Lemma 4.5 (Idempotent Lemma). *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that*

$$\mathrm{dcw}(\sigma_1 \cdots \sigma_n) \leqslant f(\max_{i \in [n]} \mathrm{dcw}(\sigma_i))$$

for every $\sigma_1, \ldots, \sigma_n$ which have the same abstraction, and this abstraction is idempotent.

Condition (1) of Simon Lemma then follows by taking f to be the maximum of the functions given by Binary and Idempotent Lemma. Thus, we are left with proving these two results. The proof of Binary Lemma is actually quite easy, while the proof of Idempotent Lemma is the main difficulty and we sketch it in the next two sections.

5 Proof of Binary Lemma

Given a clique decomposition of a graph, say of width k, and a partition $(V_1, \ldots, V_p)$ of the vertex set into p subsets, which may be non-related to the decomposition, one can adjust the decomposition at the cost of using $k \cdot p$ colors instead of k so that the final color partition of the decomposition matches $(V_1, \ldots, V_p)$. Informally, this can be done by just enriching each original label with information to which subset of $(V_1, \ldots, V_p)$ a vertex belongs. The following general-usage lemma formalizes this, and shows that the transformation may be performed by means of an MSO transduction.

Lemma 5.1 (Color Enforcement Lemma). *For all $k, p \in \mathbb{N}$ there exists a deterministic MSO transduction $\mathcal{E}_{k,p}$ with the following properties. The input vocabulary is the vocabulary of clique decompositions of width k with leaves colored using p unary predicates. The output vocabulary is the vocabulary of clique decomposition of width $k \cdot p$. Finally, on an input clique decomposition t with leaves partitioned into $(V_1, \ldots, V_p)$ using unary predicates, the output of $\mathcal{E}_{k,p}$ is a clique decomposition t' of the same graph, where in the result of t' the color of each vertex from V_i is equal to i, for all $i \in [p]$.*

Proof. The decomposition t is first adjusted to a decomposition t'' of width $k \cdot p$ with the following property: in the result of t'', the final color of every vertex is a pair consisting of its color in the result of t and the index i such that the leaf corresponding to the vertex belongs to V_i. This adjustment can be made by preserving the shape of t intact, and performing a straightforward modification to the labels of nodes. For instance, for a Join node, whenever the original label in t requested adding edges between colors c and d, the new label in t'' requests adding edges between colors (c,i) and (d,j) for all $i,j \in [p]$. Finally, we obtain t' by adding a recoloring step on top of t'' that removes the first coordinate of every color. □

Using Color Enforcement Lemma we can prove Binary Lemma.

Proof of Binary Lemma. Let $m = \max(\mathrm{dcw}(\sigma), \mathrm{dcw}(\tau))$, and let $\mathcal{D}_\sigma$ and $\mathcal{D}_\tau$ be decomposers of size at most m such that $\mathcal{D}_\sigma$ produces at least one output on the underlying graph of σ, and similarly for $\mathcal{D}_\tau$. Using coloring, we first guess the partition of the vertex set into vertices that belong to the underlying graphs of σ and τ. Next, we guess the color partition of the underlying graph of σ. Finally, for the underlying graph of τ, we guess the partition of its vertices according to profiles in τ. Note that the validity of this guess, or more precisely the fact that the adjacency between the σ-part and the τ-part depends only on the (color,profile) pair of respective vertices, can be checked using a filtering step.

We now apply $\mathcal{D}_\sigma$ to the σ-part of the graph, yielding a clique decomposition t_σ of the underlying graph of σ of width at most m. Using the origin-preserving property of decomposers (property (b) of Definition 3.1), we may assume that the color partition of σ is present on the leaves of t_σ. By applying the transduction $\mathcal{E}_{m,k}$ given by the Color Enforcement Lemma to t_σ, we may further assume that the result of the obtained decomposition t_σ has the color partition equal to the color partition of σ. Similarly, by applying $\mathcal{D}_\tau$ followed by $\mathcal{E}_{m,2^k}$ for the profile partition, we turn the τ-part of the graph into its clique decomposition t_τ whose result has the color partition equal to the profile partition in τ. Since the adjacency between the σ-part and the τ-part depends only on the (color,profile) pair of respective vertices, it now suffices to add one binary Join node, with the roots of t_σ and t_τ as children, where we request adding edges between appropriate pairs of vertices, selected by color on the σ-side and profile on the τ-side. □

6 Idempotent Lemma: sketch

In this section we sketch how to prove the Idempotent Lemma assuming a technical result called the Definable Order Lemma, which we will explain in a moment. Consider a sequence $\sigma_1, \ldots, \sigma_n$ of k-derivations such that for some abstraction e that is idempotent in T_k, we have $e = [\![\sigma_1]\!] = \ldots = [\![\sigma_n]\!]$. Let $\sigma = \sigma_1 \cdots \sigma_n$, and let G be the underlying graph of σ. Moreover, for $i \in [n]$ by G_i we denote the underlying graph of σ_i, and we call it also the *i-th block*.

Let $\preceq$ be the linear quasi-order (i.e. a total, transitive and reflexive relation) defined on the vertex set of G as follows: $u \preceq v$ holds if and only if u belongs to the i-th block and v belongs to the j-th block for some $i \leqslant j$. Similarly, let $\equiv$ be the equivalence relation on the vertex set of G defined as belonging to the same block: $u \equiv v$ iff $u \preceq v$ and $v \preceq u$. The relations $\preceq$ and $\equiv$ will be called the *block order* and the *block equivalence*, respectively. Rough idea is to show that the block order, hence also the block equivalence, can be interpreted using a bounded size (nondeterministic) MSO formula, i.e. it has bounded (in terms of k) interpretation complexity as defined below.

Definition 6.1 (Interpretation complexity)**.** Suppose that $\mathfrak{A}$ is a relational structure, and let R be a relation on its universe, say of arity n. Define the *interpretation complexity* of R inside $\mathfrak{A}$ to be the smallest m such that there exist subsets $X_1, \ldots, X_m$ of the universe in $\mathfrak{A}$ and an MSO formula $\varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ of quantifier rank at most m over the vocabulary of $\mathfrak{A}$ such that

$$(x_1, \ldots, x_n) \in R \quad \text{iff} \quad \varphi(x_1, \ldots, x_n, X_1, \ldots, X_m)$$

for all $x_1, \ldots, x_n$ in $\mathfrak{A}$.

If the interpretation complexity of the block order in G was bounded by a function of k, then we would construct a clique decomposition of G as follows: first construct clique decompositions of all blocks, and then combine them sequentially along the block order. Unfortunately, in general we cannot hope for such a bound. To see this, consider the example where G consists of, say, two disjoint paths of length n each. In this example, each σ_i introduces the i-th vertex of each of the two paths. It is not difficult to see that in this example the interpretation complexity of the block order grows with the number of blocks. However, we can define the block order on each connected component (i.e. each of the two paths) separately, and a clique decomposition of the whole graph can be obtained by putting a Join over decompositions of components. Thus, the obtained decomposition has a different shape than the input linear decomposition corresponding to the product $\sigma_1 \cdots \sigma_n$. The next statement, which is our main technical result toward the proof of Idempotent Lemma, explains how this plan can be implemented.

Lemma 6.2 (Definable Order Lemma)**.** *Let $\sigma_1, \ldots, \sigma_n$ be k-derivations as in the assumption of the Idempotent Lemma. There exists a set $Z \subseteq \binom{\mathscr{C}_k}{1,2}$ such that if $\sim$ is the relation of being in the same connected component in the Z-flip of $\sigma_1 \cdots \sigma_n$, then the relation $\sim \cap \preceq$ has interpretation complexity over G bounded by a function of k.*

The proof of the Definable Order Lemma spans most of the full version of the paper and is the technical cornerstone of our approach. In the next section we provide a short sketch of this proof, while a complete argument can be found in the full version. For the remainder of this section we sketch how Idempotent Lemma follows from Definable Order Lemma.

As argued, the idea is to interpret the block order using Definable Order Lemma, apply assumed bounded-size transductions to each block separately, thus obtaining clique decompositions of blocks, and finally to combine the obtained clique decompositions along the block order. The combination step is given by the following Combiner Lemma, whose proof follows by a careful composition of all the given pieces.

Define an *order-using decomposer* to be an MSO transduction which inputs a graph G together with a linear quasi-order on its vertices and which outputs clique decompositions of the input graph. On a given input, an order-using decomposer might produce several outputs, possibly zero.

Lemma 6.3 (Combiner Lemma, ⋆)**.** *For every $m \in \mathbb{N}$ there is an order-using decomposer $\mathcal{D}$ with the following property. Let $\tau_1, \ldots, \tau_n$ be k-derivations whose underlying graphs have definable cliquewidth at most m. Let G be the underlying graph of $\tau_1 \cdots \tau_n$ and $\preceq$ be the block order arising from product $\tau_1 \cdots \tau_n$. Then $\mathcal{D}$ produces at least one output on $(G, \preceq)$.*

With the Combiner Lemma in place, we sketch the proof of the Idempotent Lemma.

Sketch of the proof of Idempotent Lemma. Let $\sigma_1, \ldots, \sigma_n$ be the given k-derivations as in the Idempotent Lemma, i.e., with the same idempotent abstraction e. Let

$$K = \max_{i \in [n]} \mathrm{dcw}(\sigma_i).$$

Denote $\sigma = \sigma_1 \cdots \sigma_n$. Let G be the underlying graph of σ, let G_i be the underlying graph of σ_i for each $i \in [n]$, and let $\preceq$ be the block order in G compliant with the product $\sigma_1 \cdots \sigma_n$. It suffices to describe a decomposer of size bounded in terms of k and K that constructs a clique decomposition of G.

First, using coloring we enrich the structure with unary predicates that encode the partition of the vertex set of G into cells $\sigma[c]$, for $c \in \mathscr{C}_k$. Then apply the Definable Order Lemma to $\sigma_1, \ldots, \sigma_n$, yielding Z and $\sim$. Note that Z is chosen among $2^{|\binom{\mathscr{C}_k}{1,2}|}$ options, so we can nondeterministically guess Z using, say, some coloring. Having Z fixed, the equivalence relation $\sim$ (being in the same connected component of the Z-flip of σ) can be added to the structure using interpretation. By the Definable Order Lemma, we can add also the relation $\preceq \cap \sim$ to the structure, as this increases the size of the transduction only by a function of k and K. Note here that the subsets $X_1, \ldots, X_m$ in the definition of interpretation complexity of $\preceq \cap \sim$ can be guessed using coloring.

The next claim says that restricting blocks to equivalence classes of $\sim$ yields graphs of bounded definable cliquewidth.

Claim 2 ($\star$). *There is $m \in \mathbb{N}$ depending only k and K such that for every equivalence class F of $\sim$ and every $i \in [n]$, the subgraph induced in G_i by vertices contained in F has definable cliquewidth at most m.*

Claim 2 seems simple: since the definable cliquewidth of the ith block is bounded by K, the subgraph induced by F within this block should also have bounded cliquewidth. However, the proof is actually far less obvious than it seems, as the decomposer for the induced subgraph needs to work on this subgraph only, so we cannot just apply the decomposer for the whole block and restrict the outcome to the vertices of F. The proof can be found in the full version and the main idea is as follows. The part of the block outside of F can be replaced by a bounded-size equivalent graph with the same MSO type (of high enough rank) so that the assumed decomposer also decomposes the modified block after the replacement. Then the decomposer for the induced subgraph may guess this bounded-size equivalent graph, run the assumed decomposer for the block, and finally remove all the guessed vertices.

Now apply Combiner Lemma to the parameter m given by Claim 2, yielding an order-using decomposer $\mathcal{D}$ satisfying:

Claim 3. *For each equivalence class F of $\sim$, the order-using decomposer $\mathcal{D}$ produces at least one output on $(G, \preceq)$ restricted to the vertices of F.*

We now sketch the remainder of the proof, which is conceptually simple, but a formal reasoning requires attention to details. The idea is to apply decomposer $\mathcal{D}$ to each equivalence class of $\sim$ in parallel. For this, we formulate an auxiliary result called *Parallel Application Lemma* which says that this is possible using a transduction $\mathcal{D}^\star$ whose size is bounded in terms of the size of $\mathcal{D}$ (notably, Parallel Application Lemma is also used in the proof of Combiner Lemma to apply decomposers to blocks in parallel). At this point, the structure contains, for each equivalence class F of $\sim$, a clique decomposition of the subgraph induced by F in G. Moreover, by the origin-preserving property of decomposers (property (b) of Definition 3.1), the leaves of these decompositions still hold information on cells from $\{\sigma[c] : c \in \mathscr{C}_k\}$ to which the corresponding vertices belong. It remains to adjust the decompositions so that this information can be read from the color of each vertex in the result of the corresponding decomposition (for this one may use Color Enforcement Lemma), and combine all these decomposition by attaching them below a new root Join node, where we request adding edges between cells of $\{\sigma[c] : c \in \mathscr{C}_k\}$ as prescribed by Z. □

7 Definable Order Lemma: sketch

In this section we give a sketch of the proof of the Definable Order Lemma; the full argument can be found in the full version. Adopting the notation from the statement, let $e = (L, \rho, \phi)$ be the common idempotent abstraction; recall that $e = [\![\sigma]\!] = [\![\sigma_1]\!] = \ldots = [\![\sigma_n]\!]$. Recall also that the underlying graphs of $\sigma_1, \ldots, \sigma_n$ are called *blocks*.

Enriching the structure. The first step is to enrich the graph G with unary predicates to facilitate defining the block order; recall that a bounded number of such nondeterministically guessed predicates can be used by the formula φ defining the block order. First, for every cell $c \in \mathscr{C}_k$ define $U_c = \bigcup_{i \in [n]} \sigma_i[c]$. Note that U_c is not necessarily equal to the $\sigma[c]$, because the colors and profiles of vertices in σ and respective σ_i may differ. For each $c \in \mathscr{C}_k$, we have a unary predicate that selects the vertices of U_c in G.

Second, we will often end up in a situation where we know that some vertices u, v belong to blocks that are near each other, say the indices of their blocks differ by at most 3, but defining the precise block order between them is problematic. For this, we introduce *moduli*: we partition the graph into 7 parts $W_0, \ldots, W_6$ so that vertices of the i-th block belong to the part $W_{i \bmod 7}$. Observe that knowing that the block indices of u and v differ by at most 3, the relation between their blocks can be inferred from comparing their moduli.

Define structure $\widehat{G}$ to be G enriched with unary predicates selecting U_c, for all $c \in \mathscr{C}_k$, and W_r, for all $r \in \{0, 1, \ldots, 6\}$. The formula φ in the definition of interpretation complexity can have a bounded number of nondeterministically guessed unary predicates, so from now on we may work over $\widehat{G}$.

Classifying cells. Let us take a closer look on the common idempotent abstraction e. Since $e = (L, \rho, \phi)$ is idempotent in the semigroup of abstractions, it follows that ϕ is idempotent in the semigroup of functions from $[k]$ to $[k]$. This means that each color in the image of ϕ is a fixed-point of ϕ. In terms of recolorings in the sequence $\sigma_1 \cdots \sigma_n$, this implies the following: if a vertex $u \in G_s$ is colored with color i in σ_s, then the recoloring in σ_{s+1} recolors it to $\phi(i)$, and its color stays equal to $\phi(i)$ when composing with all further derivations σ_t for $t > s + 1$.

Take any pair of cells $c, d \in \mathscr{C}_k$, not necessarily different, say $c = (i, X)$ and $d = (j, Y)$. Classify the *type* of the pair (c, d) as follows:

- (c, d) is *negative* if $\phi(j) \notin X$ and $\phi(i) \notin Y$;
- (c, d) is *positive* if $\phi(j) \in X$ and $\phi(i) \in Y$; and
- (c, d) is *mixed* if $\phi(j) \in X$ and $\phi(i) \notin Y$, or $\phi(j) \notin X$ and $\phi(i) \in Y$.

It can be easily inferred from the idempotence of recolorings that if $u \in U_c$ and $v \in U_d$, and u, v belong to blocks that are neither same nor next to each other, then

- u and v are non-adjacent if (c, d) is negative,
- u and v are adjacent if (c, d) is positive, and
- if (c, d) is mixed, then u and v are adjacent if and only if the block of u is before the block of v.

Therefore, within negative and positive pairs of cells the (co-)adjacency is only local—between neighboring blocks—while for mixed pairs, the adjacency roughly defines the block order. With the help of moduli we can now prove the following lemma. Here, a cell c is *essential* if $c \in L$, that is, it is essential in the abstraction e. Equivalently, $\sigma_i[c]$ is non-empty for every $i \in [n]$.

Lemma 7.1. *If (c, d) is a mixed pair of essential cells, then the interpretation complexity of the block order restricted to $U_c \cup U_d$ over $\widehat{G}$ is at most* 2.

Social graph. Lemma 7.1 in particular shows that whenever an essential cell c is in a mixed relation with some other essential cell, then the block order within U_c already has interpretation complexity 2. Unfortunately, there may be still essential cells that are not in a mixed relation with any other cell, and hence we cannot interpret the block order for them using Lemma 7.1. We call such cells *solitary*, while all the essential cells to which Lemma 7.1 can be applied are *social*. Define the *social graph* as the graph on the social cells where two cells are considered adjacent if they form a mixed pair. Vertices belonging to U_c for c being solitary/social are respectively called solitary/social.

Using Lemma 7.1, it is straightforward to extend the interpretation of the block order to any connected component of the social graph, as described next.

Lemma 7.2. *Suppose C is a connected component of the social graph. Then the interpretation complexity of the block order restricted to $\bigcup_{c \in C} U_c$ over $\widehat{G}$ is $O(|\mathscr{C}_k|)$.*

However, we still need to relate the components of the social graph to each other, if possible, and to link solitary vertices to them.

Flipping. To have a better grasp of the solitary vertices, we now perform the flip operation. Precisely, let Z be the set of all subsets $\{c, d\} \subseteq \mathscr{C}_k$ (possibly $c = d$) such that (c, d) is positive. Consider now performing Z-flip in σ. Note that a priori this may not be the same as applying a flip between U_c and U_d for each $\{c, d\} \in Z$, as a vertex can belong to a different cell in σ and in respective σ_i. However, our choice of Z actually implies that both these flip variants—between cells of σ and between sets U_c—lead to exactly the same graph; call it H. Intuitively, by flipping we obtain that H has only negative and mixed pairs of cells. In particular, if u and v respectively belong to U_c and U_d, where (c, d) is not a mixed pair, then u and v can be adjacent in H only if they belong to the same or to neighboring blocks. Note that this always happens if either u or v is solitary.

Solitary paths. This suggests the following definition of a *solitary path*: a path P in H is *solitary* if every its internal vertex is solitary and the cells c, d such that the endpoints of P belong to U_c, U_d do not form a mixed pair. Intuitively, solitary paths realize connections between the connected components of the social graph, as well as between those components and solitary vertices. By the observation of the previous paragraph, solitary paths are local: every pair of consecutive vertices on a solitary path belongs to neighboring blocks. This gives a good combinatorial grasp on them.

More precisely, using the idempotence of the abstraction e, in particular the fact that the connectivity registries in the abstractions of derivations $\sigma, \sigma_1, \ldots, \sigma_n$ are equal, we can prove the following assertions about locality of solitary paths:

(1) If two vertices u and v from the same block can be connected by a solitary path, then they can be connected by a solitary path that visits only this block and the two neighboring blocks.
(2) If a solitary vertex u can be connected to a component D of the social graph using a solitary path, then it can be connected by a solitary path that visits only the block of u and the two neighboring blocks.
(3) If two components D, D' of the social graph can be connected by a solitary path, then for each block there is a solitary path that realizes this connection and is entirely contained in this block.

These three properties together give us a way to extend the interpretation of the block order to the connected components of H. Observe that if two vertices can be connected by a path in H, then this path can be decomposed into parts that stay within components of the social graph, and solitary paths between them. The block order within the components of the social graph is already interpreted. On the other hand, properties (1)–(3) imply that the solitary paths between the social parts can be assumed to be entirely contained in three consecutive blocks. This allows existential guessing of these paths so that we maintain a good control over the change of block indices between the endpoints. More precisely, by requiring that the vertices of a guessed solitary path uses only three consecutive moduli, we require that the path in fact uses only three consecutive blocks; this follows from the property that solitary paths may jump only between neighboring blocks.

8 Conclusions

We proved that for every k there is an MSO-transduction that defines for a given graph of linear cliquewidth k a width-$f(k)$ clique decomposition of this graph. A consequence of this result is that recognizability equals CMSO$_1$-definability on graphs of bounded linear cliquewidth.

The main open question is whether our result can be generalized from linear clique decompositions to general clique decompositions. The approach used in [3] for lifting the pathwidth case to the treewidth case heavily relies on combinatorial techniques specific to tree decompositions, and hence it seems hard to translate the ideas to the setting of clique decompositions.

References

[1] I. Adler and M.M. Kanté. 2015. Linear rank-width and linear clique-width of trees. *Theor. Comput. Sci.* 589 (2015), 87–98.

[2] Mikołaj Bojańczyk, Martin Grohe, and Michał Pilipczuk. 2018. Definable decompositions for graphs of bounded linear cliquewidth. *CoRR* abs/1803.05937 (2018).

[3] Mikołaj Bojańczyk and Michał Pilipczuk. 2016. Definability equals recognizability for graphs of bounded treewidth. In *LICS 2016*. ACM, 407–416.

[4] Mikołaj Bojańczyk and Michał Pilipczuk. 2017. Optimizing Tree Decompositions in MSO. In *STACS 2017 (LIPIcs)*, Vol. 66. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 15:1–15:13.

[5] Bruno Courcelle. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.* 85, 1 (1990), 12–75.

[6] Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic — A Language-Theoretic Approach.* Encyclopedia of mathematics and its applications, Vol. 138. Cambridge University Press. I–XIV, 1–728 pages.

[7] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. 2000. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. *Theory Comput. Syst.* 33, 2 (2000), 125–150.

[8] Bruno Courcelle and Stephan Olariu. 2000. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics* 101, 1-3 (2000), 77–114.

[9] Frank Gurski and Egon Wanke. 2005. On the relationship between NLC-width and linear NLC-width. *Theor. Comput. Sci.* 347, 1-2 (2005), 76–89.

[10] P. Heggernes, D. Meister, and C. Papadopoulos. 2011. Graphs of linear clique-width at most 3. *Theor. Comput. Sci.* 412, 39 (2011), 5466–5486.

[11] P. Heggernes, D. Meister, and C. Papadopoulos. 2012. Characterising the linear clique-width of a class of graphs by forbidden induced subgraphs. *Discrete Applied Mathematics* 160, 6 (2012), 888–901.

[12] Sang-il Oum. 2005. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B* 95, 1 (2005), 79–100.

[13] Sang-il Oum and Paul D. Seymour. 2006. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B* 96, 4 (2006), 514–528.

[14] Imre Simon. 1990. Factorization Forests of Finite Height. *Theor. Comput. Sci.* 72, 1 (1990), 65–94.