Treewidth, Pathwidth and Cospan Decompositions with Applications to Graph-Accepting Tree Automata[☆]

Christoph Blume, H.J. Sander Bruggink, Martin Friedrich, Barbara König

Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften, Abteilung für Informatik und Angewandte Kognitionswissenschaft, D-47048, Duisburg, Germany

Abstract

We will revisit the categorical notion of cospan decompositions of graphs and compare it to the well-known notions of path decomposition and tree decomposition from graph theory. More specifically, we will define several types of cospan decompositions with appropriate width measures and show that these width measures coincide with pathwidth and treewidth. Such graph decompositions of small width are used to efficiently decide graph properties, for instance via graph automata. Hence we will give an application by defining graph-accepting tree automata, thus integrating previous work by Courcelle into the setting of cospan decompositions. Furthermore we will show that regardless of whether we consider path or tree decompositions, we arrive at the same notion of recognizability.

Keywords: cospans, graph decompositions, pathwidth, treewidth, tree automata

1. Introduction

In graph rewriting the notion of cospan plays a major role: cospans can be seen as graphs equipped with an inner and an outer interface and they can be used as (atomic) building blocks for constructing or decomposing larger graphs. Furthermore cospans are a means to cast graph rewriting into the setting of reactive systems [1, 2].

In graph theory there are different notions for decomposing graphs: path and tree decompositions [3], which at first glance seem to have a very different flavour than cospan decompositions. These notions lead to width measures such as pathwidth and treewidth and they are used to specify how similar a graph is to a path or a tree. Treewidth plays a major role in complexity theory: for instance Courcelle's theorem [4] states that every graph property that can be specified

[☆]Supported by the DFG-project GaReV.

Email addresses: christoph.blume@uni-due.de (Christoph Blume), sander.bruggink@uni-due.de (H.J. Sander Bruggink), martin.friedrich@stud.uni-due.de (Martin Friedrich), barbara_koenig@uni-due.de (Barbara König)

in monadic second-order graph logic can be checked in linear time on graphs of bounded treewidth. Furthermore there are intuitive game characterizations (robber and cops games) for treewidth.

In this paper we show that, when seen from the right perspective, graph decompositions based on cospans are in fact very similar to path and tree decompositions. In order to be able to state this formally we classify several types of cospan decompositions, which are sequences of cospans (with varying additional conditions). Obtaining the decomposed graph amounts to taking the colimit of the resulting diagram. We define width measures based on such decompositions and show that the width measures all coincide with pathwidth. In the second part of the paper the results are repeated for tree-like decompositions and treewidth, where the tree-like decompositions are trees where the edges are labeled with spans or cospans, and the decomposed graph is again obtained by taking the colimit.

Additionally, we define automata for such decompositions. For cospan decompositions we use automaton functors [5], which in [6] were used to check invariants of graph transformation systems. Automaton functors work by decomposing a graph into (atomic) cospans, and then running a finite automaton on the sequence. This approach is an extension of the work by Courcelle and others on recognizable graph languages [4], which are in turn equivalent to the notion of inductive graph properties [7].

For tree-like decompositions, we define consistent tree automata. These automata input so-called term decompositions, which are tree-like decompositions in the form of first-order terms. They are usual tree automata [8, 9] with the additional requirement that their behaviour and acceptance are the same for different term decompositions of the same graph. A main result of the paper is that automaton functors and consistent tree automata accept the same language class, namely the class of recognizable graph languages.

As far as we know there have been only few investigations into the notions of pathwidth and treewidth in the context of graph rewriting. We are mainly aware of the relation between context-free (or hyperedge replacement) grammars and bounded treewidth that is discussed in [10, 11, 12]. It is shown that the language generated by a context-free grammar has always bounded treewidth, that is, there is an upper bound for the treewidth of every graph in the language. This also implies the well-known result that the language of all graphs is not context-free.

Interest in the relation between tree decompositions and graph rewriting seems to have declined since, but in our opinion this area has a lot of potential for an increased interaction of graph transformation and graph theory, since graph decompositions and width measures are still of central interest to the graph theory community. As far as we are aware, the relation between cospan decompositions and tree and path decompositions has never been formally investigated and while the main ideas are fairly straightforward it turns out that there are some subtle issues to consider when translating one representation into the other. For instance, we found that there is more than one possible translation and more than one width measure.

The paper is organized as follows: In Section 2 we will introduce the preliminaries such as cospans, graph decompositions and tree automata. Then in Section 3 we will have a closer look at cospans, identifying also atomic cospans as building blocks. Then in Section 4 we will compare cospan decompositions with path decompositions and in Section 5 we will define graph automata as automaton functors for the category of cospans of graphs. In Section 6 we compare tree-like cospan decompositions with tree decompositions, and in Sections 7 and 8 we define term decompositions and tree automata operating on the them. Finally we will conclude with Section 9.

This article is based on [13]. Sections 3, 4 and 6, in which the correspondence of the various types of decomposition is discussed, correspond to that paper; Sections 5, 7 and 8, which address (tree) automata, are new.

2. Preliminaries

By $\mathbb{N}_k$ we denote the set $\{1, \ldots, k\}$. The set of finite sequences over a set A, including the empty sequence ϵ, is denoted by A^*. Composition of two sequences $\vec{a}$ and $\vec{b}$ is denoted by juxtaposition, that is by $\vec{ab}$.

If $f\colon A \to B$ is a function from A to B, we will implicitly extend it to subsets and sequences; for $A' \subseteq A$ and $\vec{a} = a_1 \ldots a_n \in A^*$: $f(A') = \{f(a) \mid a \in A'\}$ and $f(\vec{a}) = f(a_1) \ldots f(a_n)$.

2.1. Categories and Cospans

We presuppose a basic knowledge of category theory. For an arrow f from A to B we write $f\colon A \to B$ and define $\mathit{dom}(f) = A$ and $\mathit{cod}(f) = B$. For arrows $f\colon A \to B$ and $g\colon B \to C$, the composition of f and g is denoted $(f\ ;\ g)\colon A \to C$. The category **Rel** has sets as objects and relations as arrows. Its subcategory **Set** has only the functional relations (functions) as arrows.

An initial object of a category $\mathbf{C}$ is an object 0, such that for each object $K \in \mathbf{C}$ there exists a unique morphism from 0 to K, which is denoted by $!_K\colon 0 \to K$.

Let $\mathbf{C}$ be a category in which all pushouts exist. A *concrete cospan* in $\mathbf{C}$ is a pair $\langle c_L, c_R\rangle$ of $\mathbf{C}$-arrows with the same codomain: $J \xrightarrow{c_L} G \xleftarrow{c_R} K$. Two concrete cospans are isomorphic if their middle objects are isomorphic (such that the isomorphism commutes with the component morphisms of the concrete cospan). A *cospan* is an isomorphism class of concrete cospans. In the following we will confuse cospans and concrete cospans, in the sense that we represent cospans by giving a representative of the isomorphism class.

Composition of two cospans $\langle c_L, c_R\rangle, \langle d_L, d_R\rangle$ is computed by taking the pushout of the arrows c_R and d_L. Cospans are the arrows of so-called cospan categories. That is, for a category $\mathbf{C}$ with pushouts, the category $\mathit{Cospan}(\mathbf{C})$ has the same objects as $\mathbf{C}$. The isomorphism class of a cospan $c\colon J \xrightarrow{c_L} G \xleftarrow{c_R} K$ in $\mathbf{C}$ is an arrow from J to K in $\mathit{Cospan}(\mathbf{C})$ and will be denoted by $c\colon J \nrightarrow K$.

Spans are the dual notion of cospans, that is, they are (equivalence classes of) pairs of morphisms with the same domain.

Colimits can be seen as "generalized" pushouts. Given a collection (diagram) D of objects $\{A_1, \dots, A_n\}$ and morphisms between them, the *colimit of* D is an object B together with morphisms $\mu_i \colon A_i \to B$ such that the diagram commutes, and for each object B' and morphism $\mu'_i \colon A_i \to B'$ where the diagram commutes, it holds that there exists a unique $h \colon B \to B'$ such that everything commutes. We will write $\mathit{Colim}(D) = B$ in this case.

2.2. Graphs and Decompositions

A *hypergraph* over a set of labels Σ (in the following also simply called *graph*) is a structure $G = \langle V, E, \mathit{att}, \mathit{lab} \rangle$, where V is a finite set of nodes, E is a finite set of edges, $\mathit{att} \colon E \to V^*$ maps each edge to a finite sequence of nodes attached to it, and $\mathit{lab} \colon E \to \Sigma$ assigns a label to each edge. The size of the graph G, denoted $|G|$, is defined to be the cardinality of its node set, that is $|G| = |V|$. A *discrete graph* is a graph without edges; the discrete graph with node set $\mathbb{N}_k$ is denoted by D_k. We denote the *empty graph* by $\varnothing$ instead of D_0.

A graph morphism from a graph $G = \langle V_G, E_G, \mathit{att}_G, \mathit{lab}_G \rangle$ to a graph $H = \langle V_H, E_H, \mathit{att}_H, \mathit{lab}_H \rangle$ is a pair of maps $f = \langle f_V, f_E \rangle$, with $f_\mathrm{V} \colon V_G \to V_H$ and $f_\mathrm{E} \colon E_G \to E_H$, such that for all $e \in E_G$ it holds that $\mathit{lab}_G(e) = \mathit{lab}_H(f_\mathrm{E}(e))$ and $f_\mathrm{V}(\mathit{att}_G(e)) = \mathit{att}_H(f_\mathrm{E}(e))$. The category of graphs and graph morphisms is denoted by **Graph**. Recall, that the *monomorphisms* (monos) and *epimorphisms* (epis) of the category **Graph** are the injective and surjective graph morphisms, respectively. The empty graph is the initial object of **Graph**.

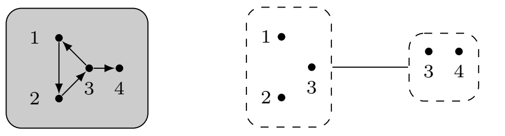
A cospan $J \xrightarrow{c_L} G \xleftarrow{c_R} K$ in **Graph** can be viewed as a graph (G) with two interfaces (J and K), called the *inner interface* and *outer interface* respectively. Informally said, only elements of G which are in the image of one of the interfaces can be "touched". By $[G]$ we denote the trivial cospan $\varnothing \to G \leftarrow \varnothing$, the graph G with two empty interfaces.

For the use in definitions we need a second kind of graph. A *simple graph* is a pair $\langle V, E \rangle$ where V is a finite set of nodes and $E \subseteq \{\{t_1, t_2\} \mid t_1, t_2 \in V, t_1 \neq t_2\}$ the set of edges.[1] In the following, v, w will range over nodes of hypergraphs, e over edges of hypergraphs, t over nodes of simple graphs and b over edges of simple graphs. In all cases, subscripts may also be used. A *tree* is a simple graph in which there exists exactly one path between each pair of nodes. A *path graph*[2] is a tree in which each node is connected to either one or two other nodes. Simple graphs, and in particular trees and path graphs, are only used to define tree and path decompositions. All objects we are decomposing will be hypergraphs.

Definition 1 (Tree decomposition). Let $G = \langle V, E, \mathit{att}, \mathit{lab} \rangle$ be a graph. A *tree decomposition* of G is a pair $\mathcal{T} = \langle T, X \rangle$, where T is a tree and $X = \{X_{t_1}, \dots, X_{t_n}\}$ is a family of subsets of V (which are called *bags* in the literature) indexed by the nodes of T, such that:

[1] Note that, by definition, a simple graph is *undirected*, *loopless* and has at most one edge between each pair of nodes.

[2] In literature, path graphs are sometimes also called *string graphs* or *paths*.



(a) Graph G_P (b) Path Decomposition of G_P

Figure 1: The graph G_P and one of its path decompositions

- for each node $v \in V$, there exists a node t of T such that $v \in X_t$;
- for each edge $e \in E$, there is a node t of T such that all nodes v attached to e are in X_t;
- for each node $v \in V$, the simple graph induced by the nodes $\{t \mid v \in X_t\}$ is a subtree of T.

The width of a tree decomposition $\mathcal{T} = \langle T, X\rangle$ is $wd(\mathcal{T}) = \big(\max_{t\in T} |X_t|\big) - 1$. A tree decomposition $\mathcal{T} = \langle T, X\rangle$ is a *path decomposition* if T is in fact a path graph.

Now, the pathwidth $pwd(G)$ and the treewidth $twd(G)$ of a graph G are defined as follows:

- $pwd(G) = \min\{wd(\mathcal{P}) \mid \mathcal{P} \text{ is a path decomposition of } G\}$,
- $twd(G) = \min\{wd(\mathcal{T}) \mid \mathcal{T} \text{ is a tree decomposition of } G\}$.

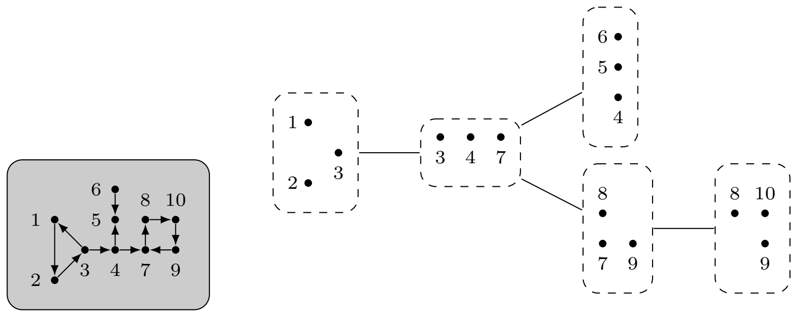
Example 1. As examples we consider only unlabeled directed graphs, that is we take $\Sigma = \{\diamond\}$ as alphabet and $|att(e)| = 2$ for every edge e. Let G_P be the graph shown in Figure 1a. Obviously, the pathwidth of this graph is 2 since it contains a 3-clique (all nodes of which have to be together in at least one bag) and we have a path decomposition $\mathcal{P}$ of width 2 which is shown in Figure 1b.

As an example for a tree decomposition we consider the unlabeled graph G_T of Figure 2a. The treewidth of this graph is 2 due to the fact that it contains a 3-clique and that the tree decomposition $\mathcal{T}$ shown in Figure 2b has width 2.

Note that the decrement in the definition of $wd(\mathcal{T})$ above is chosen so that trees have treewidth 1. Furthermore discrete graphs have pathwidth and treewidth 0 and an n-clique has both pathwidth and treewidth $n-1$ [14]. Intuitively one measures how similar a graph is to a tree or to a path. Naturally it holds that $twd(G) \leq pwd(G)$ for all graphs G, where the pathwidth might be substantially larger than the treewidth. For instance, trees can have arbitrarily large pathwidth.

2.3. Many-Sorted Terms and Tree Automata

Tree automata are a generalization of finite automata from strings to first-order terms. They are often defined in terms of algebraic structures [9] or term



(a) Graph G_T (b) Tree Decomposition of G_T

Figure 2: The graph G_T and one of its tree decompositions

rewrite systems of a certain kind [8]. Here, we give a more automata-theoretic definition. Additionally, we extend the formalism to many-sorted terms.

Let S be a set of sorts. An S-type is a pair $\langle \vec{s}, s_0 \rangle$, where $\vec{s} \in S^*$ is a sequence of *input sorts* and $s_0 \in S$ is the *output sort.* We will usually denote a type $\langle \vec{s}, s_0 \rangle$, where $\vec{s} = s_1 \dots s_n$, by $\vec{s} \to s_0$ or $\langle s_1, \dots, s_n \rangle \to s_0$, or simply by s_0 if $n = 0$.

An S-typed set M is a set M_0 together with a map $\mathit{type}\colon M_0 \to S^* \times S$ which assigns a type to each element of M. We will write $(f\colon \tau) \in M$ (or simply $f\colon \tau$ if M is clear from the context) to denote the facts that $f \in M_0$ and $\mathit{type}(f) = \tau$.

The terms we need are supposed to be *linear*, that is, every variable occurs at most once. In order to make linearity an inherent part of the definition, we use *holes*, denoted by $\Box_s$, where s is the sort of the hole, instead of named variables. For an S-typed set Σ of *function symbols* (we call Σ the *signature*) we inductively define an S-typed set $\mathcal{T}(\Sigma)$ of terms over Σ as follows:

- for $s \in S$ it holds that $(\Box_s : s \to s) \in \mathcal{T}(\Sigma)$;
- if $(f\colon \langle s_1, \dots, s_n \rangle \to s_0) \in \Sigma$ and $(t_1\colon \vec{r}_1 \to s_1), \dots, (t_n\colon \vec{r}_n \to s_n) \in \mathcal{T}(\Sigma)$, then $(f(t_1, \dots, t_n)\colon \vec{r}_1 \cdots \vec{r}_n \to s_0) \in \mathcal{T}(\Sigma)$.

Note that the second of the above inductive constructions acts as base case if $n = 0$.

If t is a term of type $\langle s_1, \dots, s_n \rangle \to s_0$ and $t_1, \dots, t_n$ are terms of types $\vec{r}_1 \to s_1, \dots, \vec{r}_n \to s_n$, respectively, then $t(t_1, \dots, t_n)$ is a term of type $\vec{r}_1 \cdots \vec{r}_n \to s_0$ which is constructed by replacing the left-most hole $\Box_{s_1}$ by t_1, the second left-most hole $\Box_{s_2}$ by t_2, etc.

Definition 2 (Tree automaton). An S-sorted (non-deterministic, bottom-up) *tree automaton* is a tuple $\mathcal{M} = \langle Q, \Sigma, \Delta, I, F \rangle$, where

- $Q = (Q_\sigma)_{\sigma \in S}$ is a family of finite sets of states indexed by S;
- Σ is an S-signature;

- $\Delta = (\Delta_f)_{f \in \Sigma}$ is a family of transition functions indexed by function symbols, where, for $f \colon \langle \sigma_1, \dots, \sigma_n \rangle \to \tau$, $\Delta_f \colon Q_{\sigma_1} \times \cdots \times Q_{\sigma_n} \to \wp(Q_\tau)$;
- $I = (I_\sigma)_{\sigma \in S}$ is a family of sets of initial states, such that $I_\sigma \subseteq Q_\sigma$ for all $\sigma \in S$; and
- $F = (F_\sigma)_{\sigma \in S}$ is a family of sets of accepting states, such that $F_\sigma \subseteq Q_\sigma$ for all $\sigma \in S$.

We define $\widehat{\Delta} = (\widehat{\Delta}_t)_{t \in \mathcal{T}(\Sigma)}$ as a family of transition functions indexed by terms, such that, for $t \colon \langle s_1, \dots, s_m \rangle \to s_0$,

$$\widehat{\Delta}_t \colon \wp(Q_{s_1}) \times \cdots \times \wp(Q_{s_m}) \to \wp(Q_{s_0})$$

Let $\overline{\Delta}$ be defined as $\overline{\Delta}_f(S_1, \dots, S_n) = \bigcup \{\Delta_f(q_1, \dots, q_n) \mid q_1 \in S_1, \dots, q_n \in S_n\}$. Then $\widehat{\Delta}$ is defined as follows:

$$\widehat{\Delta}_{\Box_s}(S) = S$$
$$\widehat{\Delta}_{f(t_1,\dots,t_n)}(S_1, \dots, S_m) = \overline{\Delta}_f(\widehat{\Delta}_{t_1}(\vec{U}_1), \dots, \widehat{\Delta}_{t_n}(\vec{U}_n)),$$

where $\vec{U}_1 \dots \vec{U}_n = S_1 \dots S_m$ and the length of each $\vec{U}_i$ is the same as the number of arguments required by $\widehat{\Delta}_{t_i}$.

A term $t \colon \langle s_1, \dots, s_m \rangle \to s_0$ is accepted by $\mathcal{M}$, if $\widehat{\Delta}_t(I_{s_1}, \dots, I_{s_m}) \cap F_{s_0} \neq \varnothing$. The *language* of $\mathcal{M}$, denoted $\mathrm{L}(\mathcal{M})$, is the set of all terms accepted by $\mathcal{M}$.

3. Cospans as Building Blocks for Graphs

Cospans of graphs can be viewed as operations on graphs with interfaces (in the sense of Courcelle [4, 15]). Let G be a graph with external nodes, as defined in [4] – which itself can be represented by a cospan $g \colon \varnothing \to G \leftarrow I$, where the interface I represents the external nodes – and let $c \colon I \to H \leftarrow K$ be a cospan. By composing g and c we obtain a cospan $(g\ ;\ c) \colon \varnothing \to GH \leftarrow K$, where GH is the pushout object of $G \leftarrow I \to H$. Recall, that taking a pushout in the category of graphs amounts to constructing the disjoint union of G and H, and subsequently fusing just enough nodes and edges to make the pushout diagram commute. That is, by composing with a cospan we can *add* new nodes and edges, *fuse* existing nodes and *change* the interface of a graph.

There exists a finite set of cospans (called *atomic cospans*) from which, together with disjoint union, all graphs with interfaces can be built; see for example [16] and [17]. Since we do not have disjoint union, we have to settle for finitely many atomic cospans *per pair of inner and outer interface* (of which there are infinitely many). Here, we use the following atomic cospans. Let $n \in \mathbb{N}$ be the size of the inner interface.

– Add a node: $\mathit{vertex}_k^n \colon D_n \leftrightarrow D_{n+1}$, where $1 \leq k \leq n+1$. This cospan is defined as:

$$\mathit{vertex}_k^n = D_n \xrightarrow{\phi} D_{n+1} \xleftarrow{id} D_{n+1}, \quad \text{where } \phi(x) = \begin{cases} x & \text{if } x < k \\ x+1 & \text{if } x \geq k. \end{cases}$$

– Remove a node from the interface: $\mathit{res}^n_k\colon D_n \nrightarrow D_{n-1}$, where $n \geq 1$ and $1 \leq k \leq n$. This cospan is defined as

$$\mathit{res}^n_k = D_n \xrightarrow{id} D_n \xleftarrow{\phi} D_{n-1}, \quad \text{where } \phi(x) = \begin{cases} x & \text{if } x < k \\ x+1 & \text{if } x \geq k. \end{cases}$$

– Add an edge: $\mathit{connect}^n_{A,\theta}\colon D_n \nrightarrow D_n$, where $A \in \Sigma$ is a label and $\theta\colon \mathbb{N}_{ar(A)} \to \mathbb{N}_n$ is a function which specifies how the new edge is connected to the nodes in the interface. This cospan is defined as

$$\mathit{connect}^n_{A,\theta} = D_n \xrightarrow{id'} G \xleftarrow{id'} D_n,$$

where $G = \langle V, E, \mathit{att}, \mathit{lab}\rangle$ with $V = \mathbb{N}_n$, $E = \{e\}$, $\mathit{att}(e) = \theta(1)\dots\theta(ar(A))$ and $\mathit{lab}(e) = A$; and $id'(x) = x$ for $x \leq n$.

– Permute the order of the nodes in the interface: $\mathit{perm}^n_\pi\colon D_n \nrightarrow D_n$. This cospan is defined as

$$\mathit{perm}^n_\pi = D_n \xrightarrow{id} D_n \xleftarrow{\pi} D_n,$$

where $\pi\colon \mathbb{N}_n \to \mathbb{N}_n$ is a permutation (that is, it is bijective).[3]

The atomic cospans are graphically depicted in Figure 3. In the following, we will use the convention that interfaces will be depicted as white-filled rectangles and center graphs will be depicted as gray-filled rectangles. Note that, for each atomic cospan $c = J \xrightarrow{c_L} G \xleftarrow{c_R} K$, the cospan $c' = K \xrightarrow{c_R} G \xleftarrow{c_L} J$, which is obtained by "flipping" c, is also an atomic cospan: the flipped version of vertex^n_k is res^{n+1}_k and vice versa, flipping $\mathit{connect}^n_{A,\theta}$ has no effect and flipping perm^n_π results in $\mathit{perm}^n_{\pi^{-1}}$.

Lemma 1. *Let $c = J \xrightarrow{c_\mathrm{L}} G \xleftarrow{c_\mathrm{R}} K$ be a cospan such that J, K are discrete and $c_\mathrm{L}, c_\mathrm{R}$ are monos. Then there exist atomic cospans $a_1, \dots, a_n$ such that $c = a_1 \,;\, \cdots \,;\, a_n$.*

In fact, there exist such atomic cospans $a_1, \dots, a_n$ such that the following condition holds: Let $a_i = I_{i-1} \to H_i \leftarrow I_i$, for $1 \leq i \leq n$. It holds that $|I_i| \leq |G|$ for all $0 \leq i \leq n$ and $|H_i| \leq |G|$ for all $1 \leq i \leq n$.

Proof. Let $J = D_k$, $K = D_m$ and $G = \langle V, E, \mathit{att}, \mathit{lab}\rangle$. We can assume without loss of generality that $V_G = \mathbb{N}_{|G|}$ and $c_\mathrm{L}(v) = v$ for every node v of D_k. Assume furthermore that $E = \{e_1, \dots, e_n\}$ and define $A_i = \mathit{lab}(e_i)$.

[3]It suffices to consider only the permutations that swap two nodes, but we do not need this restriction later on.

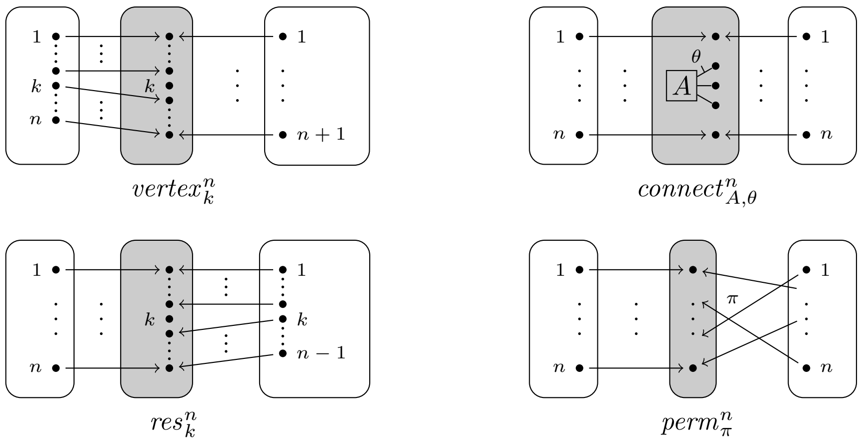


Figure 3: Graphical representations of the atomic cospans.

We construct c using the following atomic cospans:

$vertex^k_k$; $vertex^{k+1}_{k+1}$; $\cdots$; $vertex^{\|V\|-1}_{\|V\|-1}$;	add enough nodes to the graph
$connect^{\|V\|}_{A_1,\theta_1}$; $\cdots$; $connect^{\|V\|}_{A_n,\theta_n}$;	connect nodes with edges
$perm^{\|V\|}_\pi$;	move nodes of outer interface to the front
$res^{\|V\|}_{\|V\|}$; $res^{\|V\|-1}_{\|V\|-1}$; $\cdots$; $res^{\|V\|-m+1}_{\|V\|-m+1}$	remove appropriate nodes

where the θ_i are functions $\theta_i \colon \{1, \dots, |att(e_i)|\} \to \{1, \dots, |V|\}$ where $\theta_i(j)$ returns the j-th node attached to edge e_i. Furthermore π is a bijection on $\{1, \dots, |V|\}$ with $\pi(j) = f(j)$ for $1 \leq j \leq m$ and is arbitrary otherwise.

The second condition of the lemma also holds for the above atomic cospans. □

4. Path-like Decompositions: Cospan Decompositions

In this section we explore "path-like" cospan decompositions of graphs. Such decompositions are naturally defined as sequences of cospans, which are composed to a graph by taking the colimit of the emerging diagram. Equivalently, the cospans can be iteratively composed into a single cospan, where finally the interfaces are ignored.

Definition 3 (Cospan decomposition). Let G be a graph and $\vec{c} = c_1, \dots, c_n$ be a sequence of composable cospans in the category **Graph**. The sequence $\vec{c}$ is a *cospan decomposition* of G, if $Colim(\vec{c}) = G$.

Note that we now have three related notions: cospan decompositions, which are sequences of cospans; the single cospan ("graph with interfaces") which is the result of composing the cospans in a cospan decomposition; and the center graph of this cospan.

We consider the following types of cospan decompositions. The first two correspond to path decompositions in two different ways: in graph-bag decompositions the center graphs in the cospans correspond to the bags of Definition 1, whereas in interface-bag decompositions the interfaces play the role of bags. In order to make the relation between path and cospan decompositions clearer, we will only consider decompositions into cospans of injective morphisms in this paper.

Definition 4 (Type of cospan decomposition). Let $\vec{c}$ be a cospan decomposition of the graph G.

(i) $\vec{c}$ is a *graph-bag decomposition*, if all cospans have discrete interfaces and consists of injective morphisms.

(ii) $\vec{c}$ is an *interface-bag decomposition*, if it is a graph-bag decomposition, consist of pairs of jointly node-surjective morphisms[4] and it holds for all edges e of G, with $att(e) = v_1 \dots v_m$, that $v_1, \dots, v_m$ occur together in some interface.[5]

(iii) $\vec{c}$ is an *atomic cospan decomposition*, if it consists only of atomic cospans.

It is clear that the various types of cospan decomposition are strictly contained in one another, that is:

$$\text{Atomic} \subset \text{Interface-bag} \subset \text{Graph-bag} \subset \text{All}.$$

Definition 5 (Graph-bag size, interface-bag size). Let $c\colon J \to G \leftarrow K$ be a cospan. We define the *graph-bag size* and *interface-bag size* of c as follows:

$$|c|_{\text{gb}} := |V_G|$$
$$|c|_{\text{ib}} := \max\{|V_J|, |V_K|\}$$

Observe, that for all atomic cospans c it holds that $|c|_{\text{gb}} = |c|_{\text{ib}}$. For convenience later on, we define $|c|_{\text{at}} := |c|_{\text{gb}}$ ($= |c|_{\text{ib}}$).

Now we are ready to define, for all three types of cospan decomposition, a *width*:

Definition 6 (Width of cospan decomposition).

[4]Two morphisms $f\colon A \to G$ and $g\colon B \to G$ are *jointly node-surjective*, if each node of G has a pre-image in A or B (along f or g, respectively).

[5]More formally: let $I_1, \dots, I_n$ be the interfaces of the cospans in $\vec{c}$ and let $f_j : I_j \to G$ be the morphisms generated by the colimit. Then there exists an index j and nodes $w_1, \dots, w_m$ in I_j such that $f_j(w_i) = v_i$ for $i \in \{1, \dots, m\}$.

- Let $\vec{c} = c_1, \ldots, c_n$ be a cospan decomposition. We define the *graph-bag* and *interface-bag width* of $\vec{c}$ as follows:

$$wd_{\mathrm{gb}}(\vec{c}) := \max\{|c_i|_{\mathrm{gb}} \mid 1 \leq i \leq n\} - 1$$
$$wd_{\mathrm{ib}}(\vec{c}) := \max\{|c_i|_{\mathrm{ib}} \mid 1 \leq i \leq n\} - 1$$

- Let G be a graph. The graph-bag ($\mathit{cpwd}_{\mathrm{gb}}(\vec{c})$), interface-bag ($\mathit{cpwd}_{\mathrm{ib}}(\vec{c})$) and atomic cospan width ($\mathit{cpwd}_{\mathrm{at}}(\vec{c})$) of G are defined as:

$$\mathit{cpwd}_{\mathrm{gb}}(G) := \min\{wd_{\mathrm{gb}}(\vec{c}) \mid \vec{c} \text{ is a graph-bag decomposition of } G\}$$
$$\mathit{cpwd}_{\mathrm{ib}}(G) := \min\{wd_{\mathrm{ib}}(\vec{c}) \mid \vec{c} \text{ is an interface-bag decomposition of } G\}$$
$$\mathit{cpwd}_{\mathrm{at}}(G) := \min\{wd_{\mathrm{ib}}(\vec{c}) \mid \vec{c} \text{ is an atomic cospan decomposition of } G\}$$

The main theorem of this section is that, for a given graph, the three notions of cospan pathwidth are the same, and moreover are the same as the pathwidth of the graph. First, we show how to transform (cospan) path decompositions into each other:

Lemma 2.

(i) Let $\mathcal{P}$ be a path decomposition of a graph G. There exists a graph-bag decomposition $\vec{c}$ of G such that $wd_{\mathrm{gb}}(\vec{c}) = wd(\mathcal{P})$.

(ii) Let $\vec{c}$ be a graph-bag decomposition of G. There exists an interface-bag decomposition $\vec{d}$ of G such that $wd_{\mathrm{ib}}(\vec{d}) = wd_{\mathrm{gb}}(\vec{c})$.

(iii) Let $\vec{c}$ be a graph-bag decomposition of G. There exists an atomic cospan decomposition $\vec{d}$ of G such that $wd_{\mathrm{at}}(\vec{d}) \leq wd_{\mathrm{gb}}(\vec{c})$.

(iv) Let $\vec{c}$ be an interface-bag decomposition of G. There exists a path decomposition $\mathcal{P}$ of G such that $wd(\mathcal{P}) = wd_{\mathrm{ib}}(\vec{c})$.

Proof.

(i) Let $\mathcal{P} = \langle P, X\rangle$, with $P = 1 - \cdots - n$ and $G = \langle V, E, \mathit{att}, \mathit{lab}\rangle$. We construct the cospan path decomposition $\vec{c} = c_1, \ldots, c_n$ (where $c_i = J_{i-1} \to G_i \leftarrow J_i$, for $1 \leq i \leq n$) as follows:

Let $G_i = \langle V_i, E_i, \mathit{att}_i, \mathit{lab}_i\rangle$ be the graph which contains the nodes in X_i and all edges of G which are connected only to nodes of X_i and are not in some G_j, with $j < i$. Furthermore, let $J_0 := \varnothing$ and $J_n = \varnothing$ and, for $1 \leq i < n$, let J_i be the discrete graph with node set $V_i \cap V_{i+1}$.

We claim that $\vec{c}$ is a graph-bag cospan path decomposition of G with $wd_{\mathrm{gb}}(\vec{c}) = wd(\mathcal{P})$. By construction, there is an injection from every J_i and G_i into G. Moreover, since $\mathcal{P}$ is a path decomposition of G, every node of G appears in at least one G_i, while every edge appears in exactly one G_i. Also, since the bags containing a node form a subpath, nodes that appear in more than one G_i will be fused. Thus, $\mathit{Colim}(\vec{c}) = G$.

Also by construction, each G_i corresponds to some bag X_i. Thus, $wd(\mathcal{P}) = wd_{\mathrm{gb}}(\vec{c})$ directly follows.

(ii) Define, for a cospan $c = J \xrightarrow{c_\mathrm{L}} G \xleftarrow{c_\mathrm{R}} K$, the pair of cospans $\hat{c}, \check{c}$, where

- $\hat{c} = J \xrightarrow{c'_\mathrm{L}} G^- \xleftarrow{id} G^-$ and
- $\check{c} = G^- \xrightarrow{id'} G \xleftarrow{c_\mathrm{R}} K$.

where G^- is the discrete graph with the same node set as G and c'_L and id' describe the same mapping as c_L and id, respectively, but have different codomains. We observe that $\hat{c} ; \check{c} = c$.

Let $\vec{c} = c_1, \dots, c_n$. We define $\vec{d} := \hat{c_1}, \check{c_1}, \dots, \hat{c_n}, \check{c_n}$. By the observation in the previous paragraph, it holds that $\mathit{Colim}(\vec{c}) = \mathit{Colim}(\vec{d})$.

By construction, one interface of both $\hat{c}$ and $\check{c}$ consists of the nodes of G. Therefore, both $\hat{c}$ and $\check{c}$ are jointly node-surjective, all nodes of each edge occur together in some interface, and $|\hat{c}|_\mathrm{ib} = |\check{c}|_\mathrm{ib} = |c|_\mathrm{gb}$. From this it follows, that $\vec{d}$ is an interface-bag decomposition and $wd_\mathrm{ib}(\vec{d}) = wd_\mathrm{gb}(\vec{c})$.

(iii) Let $\vec{c} = c_1, \dots, c_n$. By Lemma 1, there exist, for $1 \leq i \leq n$, atomic cospan decompositions $\vec{a_i} = a_{i,1}, \dots, a_{i,m_i}$ such that $a_{i,1} ; \cdots ; a_{i,m_i} = c_i$. Define:

$$\vec{d} = a_{1,1}, \dots, a_{1,m_1}, \dots, a_{n,1}, \dots, a_{n,m_n}.$$

It follows directly from the previous observations that $\mathit{Colim}(\vec{d}) = \mathit{Colim}(\vec{c})$. From the second part of Lemma 1 it follows that $wd_\mathrm{at}(\vec{d}) \leq wd_\mathrm{gb}(\vec{c})$.

(iv) Let $\vec{c} = c_1, \dots, c_n$, where $c_i = J_{i-1} \to G_i \leftarrow J_i$. By assumption, $\mathit{Colim}(\vec{c}) = G$. Let $f_i \colon J_i \to G$ (for $0 \leq i \leq n$) and $g_i \colon G_i \to G$ (for $1 \leq i \leq n$) be the morphisms given by the colimit construction. We construct the path decomposition $\mathcal{P} = \langle P, X \rangle$ as follows: $P = 0 - \cdots - n$, with $X_i = f_i(V_{J_i})$, for $0 \leq i \leq n$.

This is a path decomposition by the following arguments: First of all, since $\mathit{Colim}(\vec{c}) = G$ and all cospans are jointly node-surjective, all nodes of G must have a pre-image in some interface J_i along the morphism f_i, and therefore appear in the bag X_i. Since $\vec{c}$ is an interface-bag decomposition, (the pre-images of) all nodes connected to a single edge must appear together in some interface and therefore the nodes must appear together in some bag. Finally, suppose there is a node v of G and bags X_p, X_q and X_r with $p < q < r$, such that v has a pre-image in X_p and X_r (over f_p and f_r, respectively). Since the colimit construction on graphs takes the disjoint union and then factors through the smallest equivalence relation which equates nodes that have a common pre-image, it must be the case that X_q contains a pre-image of v (along f_q).

From the facts that, by construction, $X_i = f_i(V_{J_i})$, and all f_i are injective, it follows that $wd(\mathcal{P}) = wd_\mathrm{ib}(\vec{c})$. □

Theorem 3. *For every graph G,*

$$pwd(G) = cpwd_\mathrm{gb}(G) = cpwd_\mathrm{ib}(G) = cpwd_\mathrm{at}(G).$$

Proof. First of all, because atomic cospan decompositions are also graph-bag and interface-bag cospan decompositions, and it easy to check that for an atomic cospan decomposition $\vec{c}$, $wd_{\mathrm{gb}}(\vec{c}) = wd_{\mathrm{ib}}(\vec{c})$, it follows for all graphs G that

$$cpwd_{\mathrm{gb}}(G) \leq cpwd_{\mathrm{at}}(G) \text{ and } cpwd_{\mathrm{ib}}(G) \leq cpwd_{\mathrm{at}}(G).$$

Together with Lemma 2 (iii) it follows that

$$cpwd_{\mathrm{gb}}(G) \geq cpwd_{\mathrm{at}}(G) \geq cpwd_{\mathrm{gb}}(G). \tag{1}$$

From Lemma 2 (i), (ii) and (iv) it follows that

$$pwd(G) \geq cpwd_{\mathrm{gb}}(G) \geq cpwd_{\mathrm{ib}}(G) \geq pwd(G). \tag{2}$$

The theorem follows directly from equations (1) and (2). □

Example 2. As an example we take the graph G_P and the corresponding path decomposition $\mathcal{P}$ of Example 1. We use the path decomposition to construct a graph-bag decomposition of G_P. For each of the two bags in $\mathcal{P}$ we take a cospan where the center graph of the first cospan is the 3-clique and the center graph of the second cospan contains the edge from the third to the fourth node. The inner interface of the first cospan and the outer interfaces of the second cospan are both empty graphs, while the outer interface of the first cospan (which is the inner interface of the second cospan) contains the third node which is the intersection of both subgraphs. The resulting graph-bag decomposition is depicted in Figure 4a. The graph-bag width of G_P is 2, since the resulting graph-bag decomposition has graph-bag size 2, and the graph-bag size of every other graph-bag decomposition must have at least size 2 due to the 3-clique which has to be contained in at least one center graph.

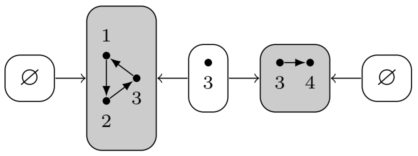
An interface-bag decomposition for the same graph is shown in Figure 4b. Note that it indeed satisfies the conditions of Definition 4: specifically each cospan is jointly node-surjective and all nodes attached to an edge live together in at least one bag. The interface-bag width of G_P is 2, due to the fact that the given interface-bag decomposition has interface-bag width 2 and any other interface-bag decomposition has to contain the nodes of the 3-clique in at least one interface.

Please note that in both cases, the graph-bag and the interface-bag decomposition, the bags of each decomposition (the center graphs in the first case and the interfaces in the second case) correspond to the bags of the path decomposition of G_P.

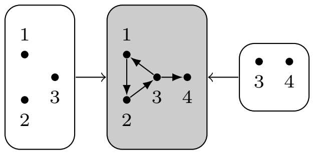
To construct the atomic decomposition we decompose the cospans of the graph-bag decomposition into atomic cospans. This is possible due to Lemma 1:

$$\begin{aligned}&vertex^0_1 \,;\, vertex^1_2 \,;\, vertex^2_3 \,;\, connect^3_{12} \,;\, connect^3_{13} \,;\, connect^3_{23} \,;\\ &\qquad res^3_0 \,;\, res^2_0 \,;\, vertex^1_2 \,;\, connect^2_{12} \,;\, res^2_0 \,;\, res^1_0,\end{aligned}$$

where we write $connect^n_{ij}$ for $connect^n_{\diamond,\theta}$ with $\theta(1) = i, \theta(2) = j$.



(a) Graph-bag decomposition of G_P



(b) Interface-bag decomposition of G_P

Figure 4: Graph-bag and interface-bag decomposition of G_P

5. Automata for Cospan Decompositions

In [5] an automaton model operating on decompositions in a general categorical setting was defined: *automaton functors*. In order to compare the language class accepted by them with the language class accepted by graph-accepting tree automata later on, in this section we briefly recapitulate this notion, instantiated to the category of cospans of graphs (we will call an automaton functor in this category a *graph automaton*).

Definition 7 (Graph automaton). A graph automaton is a structure $\mathcal{A} = \langle \mathcal{A}_0, I, F \rangle$, where

- $\mathcal{A}_0\colon \mathit{Cospan}(\mathbf{Graph}) \to \mathbf{Rel}$ is a functor which maps every graph G to a finite set $\mathcal{A}_0(G)$ (the *state set of* G) and every cospan $c\colon G \leftrightarrow H$ to a relation $\mathcal{A}_0(c) \subseteq \mathcal{A}_0(G) \times \mathcal{A}_0(H)$ (the *transition function* of c),
- $I \subseteq \mathcal{A}_0(\varnothing)$ is the set of initial states and
- $F \subseteq \mathcal{A}_0(\varnothing)$ is the set of final states.

For a graph G or a cospan c we will, in the following, usually write $\mathcal{A}(G)$ and $\mathcal{A}(c)$ instead of $\mathcal{A}_0(G)$ and $\mathcal{A}_0(c)$, respectively.

A cospan $c\colon \varnothing \leftrightarrow \varnothing$ is accepted by $\mathcal{A}$, if $\langle q, q'\rangle \in \mathcal{A}(c)$ for some $q \in I$ and $q' \in F$. The language accepted by $\mathcal{A}$, denoted by $\mathrm{L}(\mathcal{A})$, contains exactly the cospans accepted by $\mathcal{A}$. The graph language accepted by $\mathcal{A}$ is defined as

$$\mathrm{G}(\mathcal{A}) = \big\{ \mathit{Colim}(c) \mid c \in \mathrm{L}(\mathcal{A}) \big\}.$$

The intuition behind the definition is to have a mapping into a finite domain that respects compositionality and identities, that is, which is a functor. Note that, if we replace the category $\mathit{Cospan}(\mathbf{Graph})$ by the monoid over an alphabet

Σ, the above definition exactly corresponds to finite (non-deterministic) word automata. Whereas words have canonical decompositions into atomic words (letters), this is not the case for graphs. The functor property ensures that decomposing a cospan in different ways does not affect acceptance.

Note that proving that a given structure has the functor property is often non-trivial. For a useful implementation we would need an input language that enables the user to specify correct graph automata easily. As a first step in this direction, in [18] a translation from logic formulas to graph automata was defined.

The *recognizable languages* are those graph languages which are accepted by a graph automaton. Due to a result of [5] the notion of recognizability does not change if we restrict *Cospan*(**Graph**) to discrete objects and cospans with injective legs, that is, if we consider only nodes in the interface and those nodes are mapped injectively to the center graph. (Hence each node is offered to the environment only once.) In the following we will work in this restricted setting.

Example 3 (k-colorability). Let G be a graph. A k-coloring of G is a function $f\colon V_G \to \mathbb{N}_k$ such that for all $e \in E_G$ and for all $v_1, v_2 \in att_G(e)$ it holds that $f(v_1) \neq f(v_2)$ if $v_1 \neq v_2$. The following graph automaton $\mathcal{C} = \langle \mathcal{C}_0, I, F\rangle$ recognizes the k-colorable graphs:

- Every discrete graph J is mapped to $\mathcal{C}(J)$, the set of all valid k-colorings of J. Since J is discrete, this amounts to the entire function space from V_J to $\mathbb{N}_k$: $\mathcal{C}(J) = \mathbb{N}_k^{V_J}$.
- For a cospan $c\colon J \to G \leftarrow K$ the relation $\mathcal{C}(c)$ relates two colorings f_J, f_K, whenever there exists a valid coloring f for G such that $f(c_L(v)) = f_J(v)$ for every node $v \in V_J$ and $f(c_R(v)) = f_K(v)$ for every node $v \in V_K$.

Specifically we have that $\mathcal{C}(\varnothing) = \{\varnothing\}$ where $\varnothing$ is the empty coloring. Then in order to accept all k-colorable graphs with empty interfaces we take $I = F = \{\varnothing\}$: a cospan $c\colon \varnothing \mathrel{\circ\!\!\rightarrow} \varnothing$ is accepted whenever the two empty mappings are related.

The graph automaton can be understood as follows: it sequentially reads (a decomposition of) the graph. For each new node it encounters, it non-deterministically chooses a color. The graph is k-colorable if this is possible until the entire graph has been read.

In Figure 5, two states of the 3-colorability automaton functor are shown. In the first state, node 1 is colored with the first color (indicated by a circle), node 2 by the second color (indicated by a square) and node 3 by the third color (indicated by a triangle). Assume that the next symbol is $vertex_4^3$. The automaton non-deterministically goes into the next state, which has four nodes in the interface. Nodes 1–3 are colored with the same colors as before, whereas the new node can be colored by any of the colors (so there are three possible successor states). Let us assume that the new node is "colored" square. Now, let the next cospan of the input be $connect_{14}^4$. This cospan is mapped to a relation which relates the (current) state with itself, since the coloring of the current state is also a valid coloring for the successor state, because the added edge is

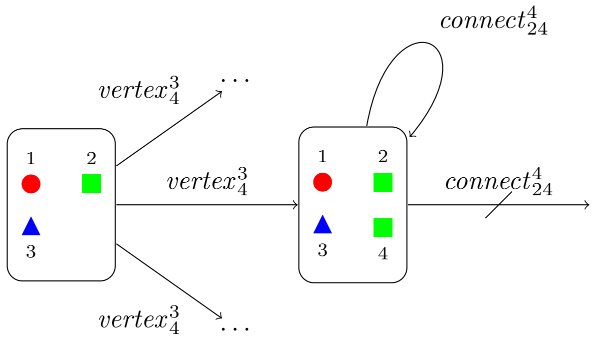


Figure 5: Example of state transitions of the 3-colorability automaton functor

incident to the first and the forth interface node which have different colors. The second input cospan, $connect_{24}^4$, is mapped to a relation which does not relate the state shown in Figure 5 to another state, since the second and the fourth interface node which are incident to the added edge, have the same color. Hence, there is no valid coloring.

When we restrict to graphs of bounded pathwidth, graph automata can be explicitly represented as a (sometimes huge) finite automaton by taking the atomic cospans as the alphabet and considering cospan decompositions as words over this alphabet. See also [19, 20].

6. Tree-like Decompositions: Star and Costar Decompositions

In this section we repeat the work of Section 4 for tree-like "cospan"-decompositions. We define *stars* and *costars* as generalizations of spans and cospans, respectively. A star $S = (\!|f_1, \ldots, f_n|\!)$ is a finite sequence[6] of morphisms with the same domain, while a costar $C = (\!|f_1, \ldots, f_n|\!)$ is a finite sequence of morphisms with the same codomain. We will consider a cospan $c\colon J \xrightarrow{c_{\mathrm{L}}} G \xleftarrow{c_{\mathrm{R}}} K$ as a special case of a costar, with $c = (\!|c_{\mathrm{R}}, c_{\mathrm{L}}|\!)$.

$$\begin{tikzcd} & G_1 & \\ & I \arrow[u] \arrow[dl] \arrow[dr] & \\ G_2 & & G_3 \end{tikzcd} \qquad \begin{tikzcd} & I_1 \arrow[d] & \\ & G & \\ I_2 \arrow[ur] & & I_3 \arrow[ul] \end{tikzcd}$$

Star Costar

Similar to cospans, costars can be seen as graphs with interfaces, of which in the case of costars there can be arbitrarily many. Intuitively, two costars can be composed over specific tentacles i and j which have the same interface K, by

[6]We define stars and costars as *sequences* of morphisms so that a) one morphism can occur more than once in the same star; and b) we can uniquely identify the tentacles of the star or costar by specifying its index.

glueing their center graphs together at K and removing the tentacles i and j. Formally:

Definition 8 (Costar composition). Let $C = \llparenthesis f_1, \dots, f_n \rrparenthesis$ be a costar with center graph $G = \mathit{cod}(f_1)$ and $D = \llparenthesis g_1, \dots, g_m \rrparenthesis$ a costar with center graph $H = \mathit{cod}(g_1)$. Furthermore, let $1 \leq i \leq n$ and $1 \leq j \leq m$ be given such that $\mathit{dom}(f_i) = \mathit{dom}(g_j)$. The composition of C and D over tentacles i and j, denoted by $C \,;_{i,j}\, D$, is defined as:

$$\begin{aligned} C \,;_{i,j}\, D = \llparenthesis & f_1 \,;\, \mu_C, \dots, f_{i-1} \,;\, \mu_C, f_{i+1} \,;\, \mu_C, \dots, f_n \,;\, \mu_C, \\ & g_1 \,;\, \mu_D, \dots, g_{j-1} \,;\, \mu_D, g_{j+1} \,;\, \mu_D, \dots, g_m \,;\, \mu_D \rrparenthesis, \end{aligned}$$

where $\mu_C \colon G \to E$ and $\mu_D \colon H \to E$ are obtained by taking the pushout of f_i and g_j, as shown in the following diagram:

$$\begin{tikzcd}
 & & K \arrow[dl, "f_i"'] \arrow[dr, "g_j"] & & \\
 \vdots \arrow[r] \arrow[r, shift left=2] & G \arrow[dr, "\mu_C"'] & (\text{PO}) & H \arrow[dl, "\mu_D"] & \vdots \arrow[l] \arrow[l, shift left=2] \\
 & & E & &
\end{tikzcd}$$

where $K = \mathit{dom}(f_i) = \mathit{dom}(g_j)$.

Note that the tentacles over which two costars are composed are hidden by the composition operation. This has the effect that the indices of the other tentacles may change in an unexpected way. For example, let $C = \llparenthesis f_1, f_2, f_3 \rrparenthesis$ and $D = \llparenthesis g_1, g_2 \rrparenthesis$. Then $C \,;_{2,1}\, D = \llparenthesis f_1 \,;\, \mu_C, f_3 \,;\, \mu_C, g_2 \,;\, \mu_D \rrparenthesis$. Here, the *second* tentacle corresponds to the *third* tentacle of C, and the *third* tentacle corresponds to the *second* of D.

We define three types of tree-like decompositions: *costar decompositions*, *star decompositions* and *atomic star decompositions*. The names of the first two relate to the form of the stars (joins) in the tree; the third one is a special case of the second. Where a cospan can be seen as a graph with two interfaces, a costar can be seen as a graph with an arbitrary number of interfaces. A costar decomposition is a decomposition into costars, where costars are connected via the interfaces in such a way that they form a tree. Note that the edges of this tree are spans. On the other hand, a star decomposition is a decomposition into stars, where the edges of the corresponding tree-like structure correspond to cospans (see also Figure 6). As in the case of cospan decompositions, we restrict our attention to injective morphisms.

Definition 9 (Costar decomposition, star decomposition).

(i) A *costar decomposition* is a tuple $\mathcal{C} = \langle T, \tau \rangle$, where T is a tree and τ is function which maps each node t of T to a graph and each edge $b = \{t_1, t_2\}$ of T to a span of injective morphisms

$$\tau(b) = \tau(t_1) \xleftarrow{\varphi_{b,t_1}} J_b \xrightarrow{\varphi_{b,t_2}} \tau(t_2).$$

A costar decomposition $\mathcal{C}$ is a costar decomposition *of* G if $\mathit{Colim}(\mathcal{C}) = G$.

(ii) A *star decomposition* is a tuple $\mathcal{S} = \langle T, \tau \rangle$, where T is a tree and τ is function which maps each node t of T to a discrete graph J and each edge $b = \{t_1, t_2\}$ to a cospan

$$\tau(b) = \tau(t_1) \to G_b \leftarrow \tau(t_2),$$

which consists of a pair of jointly node-surjective, injective morphisms.

A star decomposition $\mathcal{C}$ is a star decomposition *of* G if $\mathit{Colim}(\mathcal{C}) = G$ and additionally it holds for all edges e of G, with $\mathit{att}(e) = v_1 \dots v_m$, that $v_1, \dots, v_m$ occur together in $\tau(t)$ for some $t \in V_T$.

(iii) An *atomic star decomposition* is a star decomposition $\langle T, \tau \rangle$ such that $\tau(b)$ is an atomic cospan for all edges b of T.

In the case of cospan decompositions we had a clear hierarchy of the various decomposition types. In the case of tree-like decompositions, however, this is not the case: the sets of star and costar decompositions are not related with respect to inclusion. However, by definition, each atomic star decomposition is also a star decomposition.

Definition 10 (Width of (co)star decomposition). Let $\mathcal{S} = \langle T, \tau \rangle$ be a costar decomposition or a star decomposition. The width of $\mathcal{S}$ is defined as

$$wd_\star(\mathcal{S}) = \max_{v \in V_T} |\tau(v)| - 1.$$

Note, that Definition 10 bases the width of costar decompositions on the (non-interface) graphs they contain, while it bases the width of star decompositions on the interfaces. In both cases, however, the width of a decomposition depends on the size of the graphs that are in the image of the nodes of the tree T.

Definition 11 (Costar width, star width). Let G be a graph. The *costar width* ($\mathit{ctwd}_{\mathrm{co}\star}(G)$), *star width* ($\mathit{ctwd}_\star(G)$) and *atomic star width* ($\mathit{ctwd}_{\mathrm{at}\star}(F)$) of G are defined as follows:

$$\begin{aligned}
\mathit{ctwd}_{\mathrm{co}\star}(G) &= \min\{wd_\star(\mathcal{C}) \mid \mathcal{C} \text{ is a costar decomposition of } G\} \\
\mathit{ctwd}_\star(G) &= \min\{wd_\star(\mathcal{S}) \mid \mathcal{S} \text{ is a star decomposition of } G\} \\
\mathit{ctwd}_{\mathrm{at}\star}(G) &= \min\{wd_\star(\mathcal{S}) \mid \mathcal{S} \text{ is an atomic star decomposition of } G\}
\end{aligned}$$

A fourth possibility would be to define a kind of star decomposition, which lacks the requirement that the cospans on the edges are jointly node-surjective, but whose width is measured by the sizes of the middle graphs of the cospans instead of the middle graphs of the stars. This would result in the same notion of width. However, since this would not result in a nice direct correspondence to tree decompositions, we have left it out.

As in the previous section, the various notions defined in this section are equivalent to the notion of treewidth.

Lemma 4.

(i) *Let $\mathcal{T}$ be a tree decomposition of G. There exists a star decomposition $\mathcal{S}$ of G such that $wd_\star(\mathcal{S}) = wd(\mathcal{T})$.*

(ii) *Let $\mathcal{S}$ be a star decomposition of G. There exists a costar decomposition $\mathcal{C}$ of G such that $wd_\star(\mathcal{C}) = wd_\star(\mathcal{S})$.*

(iii) *Let $\mathcal{C}$ be a costar decomposition of G. There exists a tree decomposition $\mathcal{T}$ of G such that $wd(\mathcal{T}) = wd_\star(\mathcal{C})$.*

(iv) *Let $\mathcal{S}$ be a star decomposition of G. There exists an atomic star decomposition $\mathcal{S}'$ of G such that $wd_\star(\mathcal{S}') = wd_\star(\mathcal{S})$.*

Proof.

(i) Let $\mathcal{T} = \langle T, X\rangle$. We choose an arbitrary total ordering $<$ on the edges of T. We construct the star decomposition $\mathcal{S} = \langle T, \tau\rangle$, where T is the tree component of $\mathcal{T}$ and τ is defined as follows: For each node $t \in V_T$ of T, we define $\tau(t) = \langle X_t, \varnothing, \varnothing, \varnothing\rangle$, that is the discrete hypergraph with node set X_t. For each edge $b = \{t_1, t_2\} \in E_T$ we define $\tau(b) = \tau(t_1) \xrightarrow{id'} G_b \xleftarrow{id''} \tau(t_2)$, where G_b is the graph with node set $X_{t_1} \cup X_{t_2}$ which contains those edges of G which are not contained in (the center graph of) some cospan $\tau(b')$, where $b' < b$, and id' and id'' are the respective embeddings.

Now we need to show that $\mathcal{S}$ is a star decomposition of G. Let a graph G' and, for each node $t \in V_T$ and edge $b = \{t_1, t_2\} \in E_T$ of T, morphisms $f_t\colon \tau(t) \to G'$ and $f_b\colon G_b \to G'$ be given. We define a mediating morphism $h\colon G \to G'$.

Since the bags containing a node of G form a subtree of T, and every node of G occurs in some bag, for each node v of G the set $\{v' \in V_{G'} \mid f_x(v) = v' \text{ for some } x \in V_T \cup E_T\}$[7] must be a singleton (otherwise the diagram does not commute). Let w be the only element of the singleton. We must take $h(v) := w$ (otherwise the diagram does not commute). By construction, every edge $e \in E_G$ of G occurs in the domain of exactly one f_b. We must take $h(e) := g_b(e)$ (otherwise h is not a morphism). Now, h is the desired morphism, and it is unique, so $G = \mathit{Colim}(\mathcal{S})$.

Because the bags of $\mathcal{T}$ correspond one-to-one to (the node sets of) the graphs $\tau(t)$ of T, it is clear that $wd_\star(\mathcal{S}) = wd(\mathcal{T})$ and for all edges $e \in V_G$ it holds that the nodes adjacent to e occur together in $\tau(t)$ for some $t \in T$.

(ii) Let $\mathcal{S} = \langle T, \tau_{\mathcal{S}}\rangle$. We choose an arbitrary total ordering $<$ on the nodes of T; let $V_T = \{t_1, \dots, t_n\}$, with $t_1 < \cdots < t_n$. Let $f_t\colon \tau_{\mathcal{S}}(t) \to G$ and

[7]Note that, by construction, the node and edge sets of the graphs occurring in the tree decomposition, are subsets of node and edge sets of G, respectively. Therefore, f_x can actually be applied to v.

$f_b \colon G_b \to G$, where $\tau_{\mathcal{S}}(b) = J \to G_b \leftarrow K$, be the morphisms given by the colimit construction.

First we define a "skeleton" costar decomposition $\mathcal{B} = \langle T, \tau_{\mathcal{B}} \rangle$, where $\tau_{\mathcal{B}}$ is defined as follows:

- for all tree nodes $t \in V_T$, $\tau_{\mathcal{B}}(t) := \tau_{\mathcal{S}}(t)$;
- for all tree edges $b = \{t_1, t_2\} \in E_T$, where $\tau_{\mathcal{S}}(b) = J \xrightarrow{\phi} G \xleftarrow{\psi} K$, we define $\tau_{\mathcal{B}}(b) := J \xleftarrow{\phi'} G' \xrightarrow{\psi'} K$, where G' is obtained by taking the pullback of ϕ and ψ.

Now, we construct the final costar decomposition $\mathcal{C} = \langle T, \tau_{\mathcal{C}} \rangle$ by adding the edges to appropriate graphs. For all $t \in V_T$, $\tau_{\mathcal{C}}(t)$ is built from $\tau_{\mathcal{B}}$ by adding all edges of G of which the adjacent nodes all have a pre-image (along f_t) in $\tau_{\mathcal{C}}(t)$, but which have no pre-image in $\tau_{\mathcal{C}}(t')$ for some $t' < t$.

Since the "bags" of $\mathcal{C}$ correspond one-to-one to the "bags" of $\mathcal{S}$, $wd_\star(\mathcal{C}) = wd_\star(\mathcal{S})$. Furthermore, by construction, nodes of the graph $\tau_{\mathcal{S}}(t)$ which were mapped to the same node by the morphisms in the cospan $\tau_{\mathcal{S}}(b)$, are the image of the same node along the morphisms of the span $\tau_{\mathcal{C}}(b)$. Since all the edges of G are in the image of exactly one f_b, it must be the case that $\mathit{Colim}(\mathcal{C}) = \mathit{Colim}(\mathcal{S})$.

(iii) Let $\mathcal{C} = \langle T, \tau \rangle$. By assumption, $G = \mathit{Colim}(\mathcal{C})$. Let, for each $t \in V_T$ and $\{t_1, t_2\} \in E_T$, $f_t \colon \tau(t) \to G$ and $f_{\{t_1,t_2\}} \colon J_{\{t_1,t_2\}} \to G$ be the morphisms given by the colimit construction.

We construct the tree decomposition $\mathcal{T} = \langle T, X \rangle$, where T is the tree from $\mathcal{C}$. If $\tau(t) = H$, then we let $X_t := f_t(V_H)$. We show that the structure thus constructed is a tree decomposition of G.

First of all, since $G = \mathit{Colim}(\mathcal{C})$, every node of G must have a pre-image along some f_t, thus every node of G occurs in some bag X_t. Since interfaces are discrete, every edge of G occurs in the domain G_t of exactly one f_t, and thus the nodes adjacent to this edge occur together in X_t.

Furthermore, assume that for some node v the subgraph of bags of which v is an element do not form a subtree of T. That is, there are tree nodes t, t' and a tree node u on the path between t and t' such that $v \in X_t$, $v \in X_{t'}$ but $v \notin X_u$. Then we can show that $G \neq \mathit{Colim}(\mathcal{C})$ in a similar way as in the proof of Lemma 2 (iv).

(iv) By Lemma 1, we can transform any cospan with discrete interfaces and injective morphisms into an atomic cospan decomposition. That is, we can transform the edges of a star decomposition (labeled with cospans) into paths labeled with atomic cospans. □

Theorem 5. *For every graph G,*

$$twd(G) = ctwd_{\mathrm{co}\star}(G) = ctwd_\star(G) = ctwd_{\mathrm{at}\star}(G).$$

Proof. It follows from Lemma 4 (iv) and the fact that every atomic star decomposition is a star decomposition, that

$$ctwd_{\star}(G) = ctwd_{\mathrm{at}\star}(G). \tag{3}$$

Furthermore, from Lemma 4 (i)–(iii), the following inequalities follow:

$$twd(G) \geq ctwd_{\star}(G) \geq ctwd_{\mathrm{co}\star}(G) \geq twd(G). \tag{4}$$

The derived result follows directly from (3) and (4). □

Example 4. We consider the graph G_T and the tree decomposition $\mathcal{T}$ of Example 1. In order to construct a star decomposition of G_T, we take a cospan for each of the four edges (of the tree) of $\mathcal{T}$. The interfaces of these four cospans are the discrete graphs corresponding to the bags. The center graph of each cospan is the subgraph containing the nodes of both the inner and the outer interface of the cospan and (possibly) edges connecting these nodes. It has to be ensured that each edge occurs exactly once. This leads to the star decomposition shown in Figure 6a. Since the width of the given star decomposition has size 2 and the nodes of the 3-clique has to be contained together in at least one interface of any star decomposition, the star width of G_T is 2.

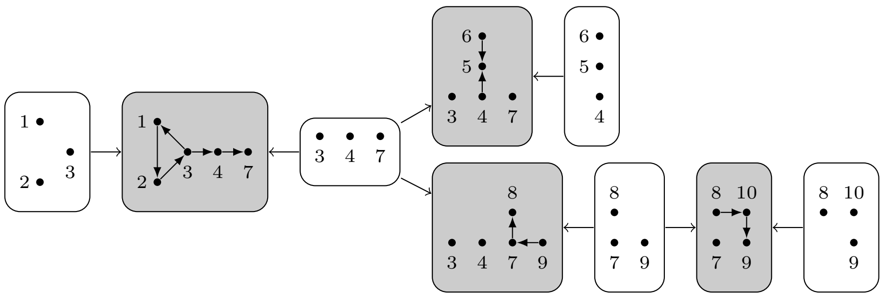
The costar decomposition can be obtained from the star decomposition. Each of the four cospans of the star decomposition is converted into a span. The inner and the outer graph of each span contain the nodes of the corresponding cospan interfaces plus additional edges. (Note that due to the conditions on star decompositions, each edge can be "shifted" into at least one interface.) The center graph of the span is then the discrete graph obtained by the intersection of the inner and the outer graphs of the span. The resulting costar decomposition is shown in Figure 6b. The costar width of G_T is 2 due to the fact that the given costar decomposition has size 2 and that any costar decomposition must contain the 3-clique in some graph of at least one span.

More details concerning the conversion of the various tree and star decompositions into each other can be found in the proof of Lemma 4.

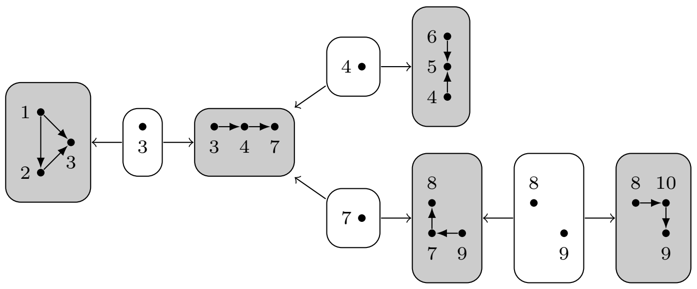
7. Term Decompositions

Our aim in the next section will be to define graph automata that operate on tree-like decompositions. On the one hand we have the tree-like decompositions of Section 6. On the other hand, however, we have tree automata which operate on *terms* rather than *trees*. Although there is a clear correspondence between trees and terms, some gaps have to be filled. That is what we do in this section, by defining *term decompositions* of graphs.

Definition 12 (Graph term). The set of sorts we employ is the set of natural numbers including zero, that is $S = \{0, 1, 2, \ldots\}$. A *graph term* is a many-sorted term over the signature $\mathcal{O}ps$, which contains (with some overloaded notation) the following function symbols:



(a) Star decomposition of G_T



(b) Costar decomposition of G_T

Figure 6: Star and costar decomposition of G_T

- $\mathit{vertex}_k^n\colon n \to n+1$, for each $n \in S$ and $1 \leq k \leq n+1$;
- $\mathit{res}_k^n\colon n \to n-1$, for each $n \geq 1 \in S$ and $1 \leq k \leq n$;
- $\mathit{connect}_{A,\theta}^n\colon n \to n$, for each label $A \in \Sigma$ and each $n \geq ar(A)$ and function $\theta\colon \mathbb{N}_{ar(A)} \to \mathbb{N}_n$;
- perm_π^n for each $n \in S$ and permutation $\pi\colon \mathbb{N}_n \to \mathbb{N}_n$;
- $\mathit{join}^n\colon \langle n, n\rangle \to n$ for each $n \in S$.

The first four of the function symbols in Definition 12 correspond to the atomic cospans; in the following we will implicitly convert between the cospans and the functions symbols. The last one plays the role of stars in star decompositions: it allows branching. Note that holes ($\square_n$) can also occur in graph terms.

We will now define how to translate graph terms into costars. Note that for a term $t\colon \langle m_1, \ldots, m_n\rangle \to m_0$ the first morphism of the costar will correspond to the root of the term (and has domain D_{m_0}), whereas the remaining n morphisms will correspond to the holes (and have domains D_{m_i}).

Definition 13 (Term decomposition of a graph).

(i) Let t be a graph term. The costar of t, denoted $\mathit{Costar}(t)$, is recursively constructed as follows:

- If $t = \Box_n$, then $\mathit{Costar}(t) = (\!| id_{D_n}, id_{D_n} |\!)$.
- If $t = f(t')$, where $f\colon m \to n$ and $t'\colon \vec{q} \to m$, then
$$\mathit{Costar}(t) = \overline{f} \;;_{2,1}\; \mathit{Costar}(t')$$
where $\overline{f}$ is the atomic cospan (see Section 3) corresponding to the function symbol f (viewed as a costar).
- If $t = \mathit{join}^n(t_1, t_2)$, then
$$\mathit{Costar}(t) = \big((\!| id_{D_n}, id_{D_n}, id_{D_n} |\!) \;;_{3,1}\; \mathit{Costar}(t_1)\big) \;;_{2,1}\; \mathit{Costar}(t_2)$$

(ii) A graph term t is a *term decomposition of a graph* G when G is the center graph of $\mathit{Costar}(t)$.

Similar to other types of decomposition we define the *width* of a term decomposition and the *term width* of a graph as follows:

Definition 14.

(i) The *width of a term decomposition* t, denoted by $\mathit{wd}(t)$, is the highest type which occurs in it minus 1; formally $\mathit{wd}(t) = \mathit{hi}(t) - 1$, where $\mathit{hi}(t)$ is inductively defined by:
$$\begin{aligned} \mathit{hi}(\Box_n) &= n & \mathit{hi}(\mathit{res}^n_k(t)) &= \max\{n, \mathit{hi}(t)\} \\ \mathit{hi}(\mathit{join}^n(t_1,t_2)) &= \max\{n, \mathit{hi}(t_1), \mathit{hi}(t_2)\} & \mathit{hi}(\mathit{connect}^n_{A,\theta}(t)) &= \max\{n, \mathit{hi}(t)\} \\ \mathit{hi}(\mathit{vertex}^n_k(t)) &= \max\{n{+}1, \mathit{hi}(t)\} & \mathit{hi}(\mathit{perm}^n_\pi(t)) &= \max\{n, \mathit{hi}(t)\} \end{aligned}$$

(ii) The *term width* of a graph G is defined as:
$$\mathit{tmwd}(G) = \min\{\mathit{wd}(t) \mid t \text{ is a term decomposition of } G\}.$$

Example 5. A term decomposition of the graph G_T of Figure 2 is the following:

```
$res^1_1(res^2_2(res^3_3(connect^3_{12}(connect^3_{23}(connect^3_{31}(vertex^2_3(vertex^1_2($
    $res^2_1(connect^2_{12}(vertex^1_2($
        $join^1($
            $res^2_2(connect^2_{12}(vertex^1_1(res^2_1(connect^2_{12}(vertex^1_1(vertex^0_1(\Box_0)))))))$
            ,
            $res^2_2(connect^2_{12}(vertex^1_1(res^2_1($
                $join^2($
                    $res^3_3(connect^3_{13}(connect^3_{32}(vertex^2_1(vertex^1_1($
                      $vertex^0_1(\Box_0))))))$
                    ,
                    $res^3_3(connect^3_{23}(connect^3_{31}(vertex^2_1(vertex^1_1($
                      $vertex^0_1(\Box_0))))))$
                $)$
            $))))$
        $)$
    $)))$
$)))))))$
```

The width of this term decomposition is 2. As in earlier examples, we write $connect^n_{ij}$ for $connect^n_{\diamond,\theta}$, where $\theta(1) = i$ and $\theta(2) = j$.

Because of the natural correspondence between trees and terms, it is no surprise that the notion of term width and the notion of atomic costar width introduced in Section 6 are equivalent.

Proposition 6. *For every graph G, $tmwd(G) = ctwd_{\mathrm{at}\star}(G)$.*

Proof (Sketch). We prove the proposition by translating every term decomposition of size k into an atomic star decomposition of size k and vice versa.

($\Rightarrow$): To translate a term decomposition into an atomic star decomposition we can proceed recursively and compose the results of the recursive calls similar to Definition 13.

($\Leftarrow$): Given an atomic star decomposition we can non-deterministically pick a root, then turn every cospan in the tree so that its outer interface is pointed towards the root (we can do this because, as observed on page 8, the inverse of each atomic cospan is also an atomic cospan) and then recursively transforming all the subtrees into a term. $\square$

8. Automata for Term Decompositions

In this section we define automata which define graph languages via their term decompositions, and we show that they accept the same language class as the graph automata defined in Section 5.

Definition 15 (Consistent tree automaton). A *consistent tree automaton* is a structure $\mathcal{M} = \langle Q, \Delta, I, F\rangle$, such that $\langle Q, \mathcal{O}ps, \Delta, I, F\rangle$ is an $\mathbb{N}$-sorted tree automaton and the following conditions apply:

- for all terms $t_1, t_2 \in \mathcal{T}(\mathcal{O}ps)$ of the same type it holds that $\widehat{\Delta}_{t_1} = \widehat{\Delta}_{t_2}$ if $Costar(t_1) = Costar(t_2)$ (*structural consistency*);
- for all term decompositions $t_1 \colon \langle 0, \dots, 0\rangle \to 0$ and $t_2 \colon \langle 0, \dots, 0\rangle \to 0$ [8] such that $Colim(Costar(t_1)) = Colim(Costar(t_2))$ it holds that $t_1 \in \mathrm{L}(\mathcal{M})$ if and only if $t_2 \in \mathrm{L}(\mathcal{M})$ (*semantic consistency*); and
- all initial and final states are in Q_0, that is, $I_k = \varnothing$ and $F_k = \varnothing$ for all $k \geq 1$.

Analogous to graph automata, the *graph language* of a consistent tree automaton $\mathcal{M}$ is defined as $\mathrm{G}(\mathcal{M}) = \{\, Colim(t) \mid t \in \mathrm{L}(\mathcal{M})\}$.

[8]Note, that the types of t_1 and t_2 *are not* necessary equal, that is the lengths of the 0-sequences may be different.

The structural consistency condition corresponds to the functor property of graph automata in Definition 7: it says that the automaton behaviour for a graph with interfaces does not depend on the specific way the graph is decomposed into a term decomposition. In the case of the graph automata the functor property was enough because all cospans have exactly two interfaces, so we only have to test whether $\varnothing \to G \leftarrow \varnothing$ is accepted. However, a graph may have term decompositions of different types, for example one of type $\langle 0, 0\rangle \to 0$ and one of type $\langle 0, 0, 0\rangle \to 0$. The semantic consistency condition is needed to make sure that also in this case the acceptance of a graph does not depend on its specific term decomposition.

As with graph automata, checking that a given structure satisfies the consistency conditions is non-trivial. Supplying building blocks that enable users to easily specify consistent tree automata is ongoing research.

Example 6. Consider the graph automaton accepting k-colorable graphs of Example 3. Let $k \in \mathbb{N}$ be given. We define the consistent tree automaton $\mathcal{M} = \langle Q, \mathcal{O}ps, \Delta, I, F\rangle$, as follows:

- $Q_n = \mathbb{N}_k^{\mathbb{N}_n}$, that is, Q_n is the set of all k-colorings of the discrete graph with n nodes.
- $I = F = Q_0$, that is, *all* states in Q_0 are both initial and final states.
- For $f \in \mathcal{O}ps \backslash \{join^n \mid n \in \mathbb{N}\}$, $\Delta_f = \mathcal{C}(c_f)$, where $\mathcal{C}$ is the graph automaton from Example 3 and c_f is the atomic cospan which corresponds to f.
- $\Delta_{join^n}(q, q) = \{q\}$ and $\Delta_{join^n}(q, q') = \varnothing$ for $q \neq q'$.

Now, $\mathcal{M}$ recognizes k-colorable graphs.

In the rest of this section we prove the following theorem, which relates the notion of consistent tree automaton to the notion of graph automaton of Definition 7.

Theorem 7. *Let L be a graph language. L is accepted by a consistent tree automaton if and only if L is accepted by a graph automaton.*

The left-to-right case of the theorem is easily proved, because (modulo some technicalities) a graph automaton can be obtained from a consistent tree automaton by 'forgetting' the transition functions for *join*. For the proof of the other direction we require some auxiliary machinery. First, we need an operation which composes two cospans in parallel by gluing over their common interface, that is, for cospans $c_1, c_2 \colon \varnothing \not\rightarrow J$, we need to glue them together over J but keep J as the interface.

Definition 16. Let $c_1 \colon \varnothing \xrightarrow{e_1} G_1 \xleftarrow{a_1} J$ and $c_2 \colon \varnothing \xrightarrow{e_2} G_2 \xleftarrow{a_2} J$ be two cospans, where J is a discrete graph. We define the cospan

$$c_1 \mathbin{//} c_2 \colon \varnothing \to G \leftarrow J = \langle e, a_1 \,;\, a_1'\rangle$$

by constructing a pushout as follows:

$$\begin{tikzcd}
& & J \arrow[dl, "a_1"'] \arrow[dr, "a_2"] & & \\
\varnothing \arrow[r, "e_1"] & G_1 \arrow[dr, "a_1'"'] & (\textsc{po}) & G_2 \arrow[dl, "a_2'"] & \varnothing \arrow[l, "e_2"'] \\
& & G & & \\
& & \varnothing \arrow[u, "e"] & &
\end{tikzcd}$$

We define, for a given graph automaton $\mathcal{A}$, the equivalence relation $\equiv_{\mathcal{A}}$ as follows: $c_1 \equiv_{\mathcal{A}} c_2$ if $\mathcal{A}(c_1) = \mathcal{A}(c_2)$.

Lemma 8. *The equivalence relation $\equiv_{\mathcal{A}}$ is a congruence, that is, if $c_1 \equiv_{\mathcal{A}} c_1'$ and $c_2 \equiv_{\mathcal{A}} c_2'$ then:*

(i) $c_1 \mathbin{;} c_2 \equiv_{\mathcal{A}} c_1' \mathbin{;} c_2'$ *and*

(ii) $c_1 \mathbin{/\!/} c_2 \equiv_{\mathcal{A}} c_1' \mathbin{/\!/} c_2'$.

Proof. Let $\mathcal{A} = \langle \mathcal{A}_0, I, F \rangle$.

(i) Follows directly from the fact that $\mathcal{A}_0$ is a functor.

(ii) Define, for a cospan $d \colon \varnothing \leftrightarrow J$ with $d = \langle d_{\mathrm{L}}, d_{\mathrm{R}} \rangle$, the cospan $\widehat{d} \colon J \leftrightarrow J$ by $\widehat{d} = \langle d_{\mathrm{R}}, d_{\mathrm{R}} \rangle$.

Consider the diagram of Definition 16. It is the case that $c_1 \mathbin{;} \widehat{c_2} = \langle e_1 \mathbin{;} a_1', a_2 \mathbin{;} a_2' \rangle$. Furthermore, $c_1 \mathbin{/\!/} c_2 = \langle e, a_1 \mathbin{;} a_1' \rangle = \langle e, a_2 \mathbin{;} a_2' \rangle$. Since the graph morphism from $\varnothing$ to G is unique, it holds that $c_1 \mathbin{/\!/} c_2 = c_1 \mathbin{;} \widehat{c_2}$. Now assume $c_1 \equiv_{\mathcal{A}} c_1'$ and $c_2 \equiv_{\mathcal{A}} c_2'$. Note also that the definition of $c_1 \mathbin{/\!/} c_2$ is completely symmetrical, and thus $c_1 \mathbin{/\!/} c_2 = c_2 \mathbin{/\!/} c_1$. Now we have:

$$\begin{aligned} c_1 \mathbin{/\!/} c_2 = c_1 \mathbin{;} \widehat{c_2} \equiv_{\mathcal{A}} c_1' \mathbin{;} \widehat{c_2} = c_1' \mathbin{/\!/} c_2 = c_2 \mathbin{/\!/} c_1' = c_2 \mathbin{;} \widehat{c_1'} \\ \equiv_{\mathcal{A}} c_2' \mathbin{;} \widehat{c_1'} = c_2' \mathbin{/\!/} c_1' = c_1' \mathbin{/\!/} c_2', \end{aligned}$$

as required. □

The translation from graph automata to consistent tree automata is carried out by the following construction. Note that it follows the same pattern as the construction of a finite automaton from the Myhill-Nerode equivalence classes of a language. In our case the equivalence is provided by the graph automaton $\mathcal{A}$ and is used to define equivalence classes of cospans of the form $\varnothing \leftrightarrow D_i$ which then serve as states. Note that for a fixed i the number of these equivalence classes is finite since $\mathcal{A}$ maps (interface) graphs to finite (state) sets and there are only finitely many relations between two given finite sets.

Definition 17. Let a graph automaton $\mathcal{A} = \langle \mathcal{A}_0, I_{\mathcal{A}}, F_{\mathcal{A}} \rangle$ be given, and let $[\![c]\!]$ denote the $\equiv_{\mathcal{A}}$-equivalence class of c. We construct the consistent tree automaton $\mathcal{M}_{\mathcal{A}} = \langle Q, \Delta, I_{\mathcal{M}}, F_{\mathcal{M}} \rangle$ with:

- $Q_i = \{[\![c]\!] \mid c\colon \varnothing \mathrel{\ooalign{$\rightarrow$}} D_i\}$.
- $I_{\mathcal{M}} = \{[\![id_\varnothing \colon \varnothing \to \varnothing]\!]\}$.
- $F_{\mathcal{M}} = \{[\![c]\!] \mid c \in \mathrm{L}(\mathcal{A})\}$.
- $\Delta_{\Box_k}([\![c]\!]) = \{[\![c]\!]\}$, for $c\colon \varnothing \to D_k$.
- For all $f \in \mathcal{O}ps \setminus \{join^n \mid n \in \mathbb{N}\}$, $\Delta_f([\![c]\!]) = \{[\![c \,;\, c_f]\!]\}$, where c_f is the cospan corresponding to the function symbol f.
- $\Delta_{join^k}([\![c_1]\!], [\![c_2]\!]) = \{[\![c_1 \mathbin{/\!/} c_2]\!]\}$, for $c_1, c_2\colon \varnothing \to D_k$.

The construction of Definition 17 is well-defined because $\equiv_{\mathcal{A}}$ is a congruence (see Lemma 8). Observe that, by construction, Δ_t is deterministic for all terms, that is, it evaluates to a singleton.

Definition 18. Let $C = (\!|g, f_1, \dots, f_n|\!)$ be a costar, where $g\colon J \to G$ and $f_i\colon K_i \to G$ (for $i \in \{1, \dots, n\}$). Furthermore, let $c_1, \dots, c_n$ be cospans with $c_i\colon \varnothing \to K_i$ (for $i \in \{1, \dots, n\}$). We define the cospan

$$C(c_1, \dots, c_n) = \varnothing \xrightarrow{!_{G'}} G' \xleftarrow{g'} J,$$

where $g'\colon J \to G'$ is the first tentacle of the composed costar

$$(\!|g', !_{G'}, \dots, !_{G'}|\!) = (\cdots (C \,;_{2,1}\, c_1) \cdots) \,;_{2,1}\, c_n.$$

Lemma 9. *Let $\mathcal{A}$ be a graph automaton and $\mathcal{M}_{\mathcal{A}} = \langle Q, \Delta, I_{\mathcal{M}}, F_{\mathcal{M}}\rangle$ a consistent tree automaton as constructed by Definition 17. Furthermore, let $t\colon \langle k_1, \dots, k_n\rangle \to k_0$ be a term decomposition and take $C_t = Costar(t)$. For all cospans $c_1, \dots, c_n$ with $c_i\colon \varnothing \to D_{k_i}$ ($i \in \{1, \dots, n\}$) it holds that*

$$\widehat{\Delta}_t([\![c_1]\!], \dots, [\![c_n]\!]) = \Big\{ [\![C_t(c_1, \dots, c_n)]\!] \Big\}.$$

Proof. By structural induction on t. □

Proposition 10. *Let $\mathcal{A}$ be a graph automaton. $\mathcal{M}_{\mathcal{A}}$ as defined in Definition 17 is a consistent tree automaton.*

Proof. Let $\mathcal{M}_{\mathcal{A}} = \langle Q, \Delta, I_{\mathcal{M}}, F_{\mathcal{M}}\rangle$. The structural consistency condition follows from the fact that, by Lemma 9, the transition function Δ_t depends only on $Costar(t)$.

To show the semantical consistency condition, observe that for costars

$$C_1 = (\!|f, \overbrace{!_G, \dots, !_G}^{n \text{ times}}|\!) \text{ and } C_2 = (\!|f, \overbrace{!_G, \dots, !_G}^{m \text{ times}}|\!)$$

where $f\colon J \to G$, it holds that $C_1(id_\varnothing, \dots, id_\varnothing) = C_2(id_\varnothing, \dots, id_\varnothing)$. Since, for term decompositions $t\colon \langle 0, \dots, 0\rangle \to 0$, the costar $Costar(t)$ is of this form, also the semantical consistency condition follows from Lemma 9. □

Now we have the necessary machinery to prove Theorem 7.

Proof (of Theorem 7). ($\Rightarrow$): Let $\mathcal{M} = \langle Q, \Delta, I, F\rangle$ be a consistent tree automaton. We observe that all function symbols of the signature except $join^n$ are unary, and therefore terms that do not contain $join^n$ are isomorphic to cospan decompositions. Thus, by Lemma 1, every graph has a term decomposition which does not contain any occurrence of $join^n$.

Therefore we can build a graph automaton from $\mathcal{M}$ by mapping each cospan c to the relation $\widehat{\Delta}(t)$, where t is a term (without $join^n$) isomorphic to c. Well-definedness of this definition and functoriality of the resulting graph automaton follow from the consistency condition on $\mathcal{M}$.

($\Leftarrow$): Let a graph automaton $\mathcal{A} = \langle \mathcal{A}_0, I, F\rangle$ be given and let $\mathcal{M}_\mathcal{A} = \langle Q, \Delta, I_\mathcal{M}, F_\mathcal{M}\rangle$ be the consistent tree automaton constructed as in Definition 17. By Proposition 10 $\mathcal{M}_\mathcal{A}$ is a consistent tree automaton which, in particular, satisfies the structural and semantic consistency conditions. It remains to show that $\mathcal{M}_\mathcal{A}$ and $\mathcal{A}$ accept the same language.

Let G be a graph, $c\colon \varnothing \leftrightarrow \varnothing$ a cospan decomposition of G and $t\colon \langle 0, \dots, 0\rangle \to 0$ a term decomposition of G. Suppose $c = c_1 \,;\, \cdots \,;\, c_n$. We define $t' = c_n(\cdots(c_1(\Box_0))\cdots)$. By construction it holds that $Costar(t') = c$ (where the cospan c is interpreted as a costar consisting of two tentacles). Because all the interfaces of $Costar(t)$ and of c are empty, and the center graph of both is G, it must hold by the semantic consistency condition that $t \in \mathrm{L}(\mathcal{M})$ if and only if $t' \in \mathrm{L}(\mathcal{M})$. By construction it holds that $\Delta_{t'}([\![id_\varnothing]\!]_\equiv) = \{[\![c]\!]_\equiv\}$. Thus:

$$t \in \mathrm{L}(\mathcal{M}_\mathcal{A}) \iff t' \in \mathrm{L}(\mathcal{M}_\mathcal{A}) \iff ([\![id_\varnothing]\!]_\equiv \in I \text{ and } [\![c]\!]_\equiv \in F) \iff c \in \mathrm{L}(\mathcal{A}),$$

as required. $\Box$

9. Conclusion

We have shown how to convert the graph-theoretical notion of path decompositions into cospan decompositions, and tree decompositions into star or costar decompositions. As we have seen there are indeed several possible choices, mainly depending on whether we identify bags with interfaces or with the center graph in a cospan. Furthermore there is in addition the notion of decomposition into atomic cospans, which can be viewed as atomic building blocks. The investigations in this paper have their origin in a Master's thesis [21].

Since the notion of tree decomposition and treewidth has many applications in graph theory, we expect that some of these applications are also useful in a more graph transformation oriented setting. As exhibited by our application we are specifically interested in using path and tree decompositions for recognizable graph languages [4] or – more specifically – for graph automata acting as acceptors of graph languages.

The idea of using (tree) automata for such purposes is not new, it has already been advocated in the work of Courcelle [22]. Our contribution here is to recast those automata in a categorical cospan setting, closer to the algebraic theory of graph rewriting. Furthermore one of our main results is that tree automata do not accept more graph languages than word automata, hence the notion of recognizability for path and tree decompositions coincides.

Still, working with tree decompositions can be profitable, since it allows to consider smaller interfaces, thus reducing the size of the state sets which grow exponentially in the width of the decomposition. On the other hand path decompositions avoid the often costly join operation and hence often have a benefit in practice.

For implementation purposes, we have to bound the width of the considered decompositions – and thus the path width of the accepted graphs – in order to obtain automata with a finite state set. We are currently working on an implementation that is based on atomic cospan decompositions, meaning that we work with (non-deterministic) finite automata which process sequences of atomic cospans (see the atomic cospan decomposition of Example 2). Furthermore we are using binary decision diagrams in order to fight state explosion.

Finally, let us remark that we did not treat the question of *how* to obtain such path or tree decompositions, given a single, monolithic graph. This is a non-trivial problem that has been studied by Bodlaender et al. [23, 24]. It can be shown that for a fixed parameter k it can be checked in linear time (in the size of the graph), whether the given graph has pathwidth or treewidth smaller than k. Furthermore the respective decompositions can be obtained in linear time. However, despite their good runtime behaviour in theory, these algorithms are not really practical, which means that heuristics are used in practice.

Acknowledgements. We thank the anonymous referees for their valuable remarks about the submitted version of this paper.

References

[1] J. J. Leifer, R. Milner, Deriving bisimulation congruences for reactive systems, in: Proceedings of CONCUR 2000, Vol. 1877 of LNCS, Springer, 2000, pp. 243–258.

[2] V. Sassone, P. Sobociński, Reactive systems over cospans, in: Proceedings of LICS '05, IEEE, 2005, pp. 311–320.

[3] N. Robertson, P. Seymour, Graph minors. II. Algorithmic aspects of tree width, Journal of Algorithms 7 (1986) 309–322.

[4] B. Courcelle, The monadic second-order logic of graphs I. Recognizable sets of finite graphs, Information and Computation 85 (1990) 12–75.

[5] H. J. S. Bruggink, B. König, On the recognizability of arrow and graph languages, in: Proceedings of ICGT '08, Vol. 5214 of LNCS, Springer, 2008, pp. 336–350.

[6] C. Blume, H. J. S. Bruggink, B. König, Recognizable graph languages for checking invariants, in: Proceedings of GT-VMT 2010, Vol. 29 of Electronic Communications of the EASST, 2010.

[7] A. Habel, H.-J. Kreowski, C. Lautemann, A comparison of compatible, finite, and inductive graph properties, Theoretical Computer Science 110 (1) (1993) 145–168.

[8] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, Available from: `http://www.grappa.univ-lille3.fr/tata`, 12 October 2007.

[9] F. Gécseg, M. Steinby, Tree Automata, Akadémiai Kiadó, Budapest, 1984.

[10] A. Habel, Hyperedge Replacement: Grammars and Languages, Vol. 643 of LNCS, Springer-Verlag, 1992.

[11] C. Lautemann, Decomposition trees: Structured graph representation and efficient algorithms, in: Proceedings of CAAP '88, Vol. 299 of LNCS, Springer, 1988, pp. 28–39.

[12] C. Lautemann, Tree automata, tree decomposition and hyperedge replacement, in: Proc. of Graph-Grammars and Their Application to Computer Science '90, LNCS, Springer, 1991, pp. 520–537, 532.

[13] C. Blume, H. Bruggink, M. Friedrich, B. König, Treewidth, pathwidth and cospan decompositions, in: Proceedings of GT-VMT 2011, Vol. 41 of Electronic Communications of the EASST, 2011.

[14] N. Robertson, P. Seymour, Graph minors. III. Planar tree-width, Journal of Combinatorial Theory, Series B 36 (1) (1984) 49–64.

[15] M. Bauderon, B. Courcelle, Graph expressions and graph rewritings, Mathematical Systems Theory 20 (1987) 83–127.

[16] F. Gadducci, R. Heckel, An inductive view of graph transformation, in: Proceedings of WADT '97, Vol. 1376 of LNCS, 1997, pp. 223–237.

[17] S. Bozapalidis, A. Kalampakas, Recognizability of graph and pattern languages, Acta Informatica 42 (8/9) (2006) 553–581.

[18] H. J. S. Bruggink, B. König, A logic on subobjects and recognizability, in: Proceedings of IFIP-TCS '10, Vol. 323 of IFIP-AICT, Springer, 2010, pp. 197–212.

[19] H. J. S. Bruggink, M. Hülsbusch, Decidability and expressiveness of finitely representable recognizable graph languages, in: Proceedings of GT-VMT 2011, Vol. 41 of Electronic Communications of the EASST, 2011.

[20] C. Blume, Efficient implementation of automaton functors for the verification of graph transformation systems, in: Proceedings of ICGT '10, Proceedings of the Doctoral Symposium, Vol. 38, Electronic Communications of the EASST, 2010.

[21] M. Friedrich, Baumautomaten und Baumzerlegungen für erkennbare Graphsprachen, Master's thesis, Universität Duisburg-Essen (July 2010).

[22] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, in: G. Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations, World Scientific, 1997, Ch. 5.

[23] H. L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (6) (1996) 1305–1317.

[24] H. L. Bodlaender, T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, Journal of Algorithms 21 (2) (1996) 358–402.