Electronic Communications of the EASST
Volume X (2011)

Proceedings of the Tenth International Workshop on Graph Transformation and Visual Modeling Techniques (GTVMT 2011)

Treewidth, Pathwidth and Cospan Decompositions

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

13 pages

Guest Editors: Fabio Gadducci, Leonardo Mariani
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Treewidth, Pathwidth and Cospan Decompositions

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

Universität Duisburg-Essen, Germany
christoph.blume@uni-due.de, sander.bruggink@uni-due.de, martin.friedrich@stud.uni-due.de,
barbara_koenig@uni-due.de

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.

Keywords: cospans, graph decompositions, pathwidth, treewidth

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 [LM00, SS05].

In graph theory there are different notions for decomposing graphs: path and tree decomposition [RS86], which at first glance seem to have a very different flavour. 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 [Cou90] states that every graph property that can be specified 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 angle, 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 compositions are trees where the edges are labeled with spans or cospans, and the decomposed graph is again obtained by taking the colimit.

Our interest in this area stems from our work on recognizable graph languages [BK08], which in [BBK10] were used to check invariants of graph transformation systems. In this context we proposed automaton functors as an automaton model for accepting graph languages. Automaton

* This work is supported by the DFG-project GaReV.

functors work by decomposing a graph into a sequence of smaller graphs with interfaces (formally cospans in the category of graphs, and therefore such sequences are called cospan decompositions), 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 [Cou90], which are equivalent to the notion of inductive graph properties [HKL93].

In order to represent such structures in a computer, the interface size of the decompositions must be bounded. We suspected for some time that this bound was strongly related to the notion of pathwidth, but the relation was never formally investigated.

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 [Hab92, Lau88, Lau91]. 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 of, 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.

In Section 2 we will introduce the preliminaries such as cospans and graph decompositions. 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 with tree decompositions. Finally we will conclude with Section 6. The proofs can be found in the full version of this paper [BBFK11].

2 Preliminaries

By $\mathbb{N}_k$ we denote the set $\{1,\ldots,k\}$. The set of finite sequences over a set A is denoted A^*. 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 $dom(f) = A$ and $cod(f) = B$. 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.

A *span* in a category $\mathbf{C}$ is a pair $\langle c_L, c_R\rangle$ of $\mathbf{C}$-arrows $G \leftarrow c_L - I - c_R \rightarrow H$. The dual notion to a span is a *cospan*. Let $\mathbf{C}$ be a category in which all pushouts exist. A cospan in $\mathbf{C}$ is a pair

$\langle c_L, c_R \rangle$ of **C**-arrows $J \xrightarrow{c_L} G \xleftarrow{c_R} K$. 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 (and spans) are isomorphic if their middle objects are isomorphic (such that the isomorphism commutes with the component morphisms of the cospan). Isomorphism classes of cospans are the arrows of so-called cospan categories. That is, for a category **C** with pushouts, the category $Cospan(\mathbf{C})$ has the same objects as **C**. The isomorphism class of a cospan $c\colon J \xrightarrow{c_L} G \xleftarrow{c_R} K$ in **C** is an arrow from J to K in $Cospan(\mathbf{C})$ and will be denoted by $c\colon J \nrightarrow K$.

Colimits can be seen as "generalized" pushouts. Given a collection (diagram) D of objects $\{A_1, \ldots, A_n\}$ and morphisms between them, the *colimit of* D is an object B together with morphisms $f_i\colon A_i \to B$ such that the diagram commutes, and for objects B' and morphisms $f'_i\colon A_i \to B'$ where the diagram commutes, it holds that there exists a unique $h\colon B \to B'$ such that the diagram commutes. We will write $Colim(D) = B$ in this case (where we allow D to be any representation of a diagram, for example a sequence).

2.2 Graphs and Decompositions

A *hypergraph* over a set of labels Σ (in the following also simply called *graph*) is a structure $G = \langle V, E, att, lab \rangle$, where V is a finite set of nodes, E is a finite set of edges, $att\colon E \to V^*$ maps each edge to a finite sequence of nodes attached to it, and $lab\colon E \to \Sigma$ 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 $\emptyset$ instead of D_0.

A graph morphism is a structure preserving map between two graphs. 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.

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 $\emptyset \to G \leftarrow \emptyset$, the graph G with two empty interfaces.

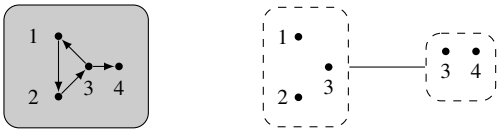
For the use in definitions we define a second kind graph. A *simple graph* is a tuple $\langle V, E \rangle$ where V is a finite set of nodes and $E \subseteq \{\{t_1, t_2\} \mid t_1, t_2 \in V, t_1 \neq t_2\}$ the set of edges.[1] A *tree* is a simple graph in which there exists exactly one path between 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 Let $G = \langle V, E, att, lab \rangle$ be a graph. A *tree decomposition* of G is a pair $\mathcal{T} = \langle T, X \rangle$, where T is a tree and $X = \{X_{t_1}, \ldots, X_{t_n}\}$ is a family of subsets of V (which are called *bags* in the literature) indexed by the nodes of T, such that:

– for each node $v \in V$, there exists a node t of T such that $v \in X_t$;

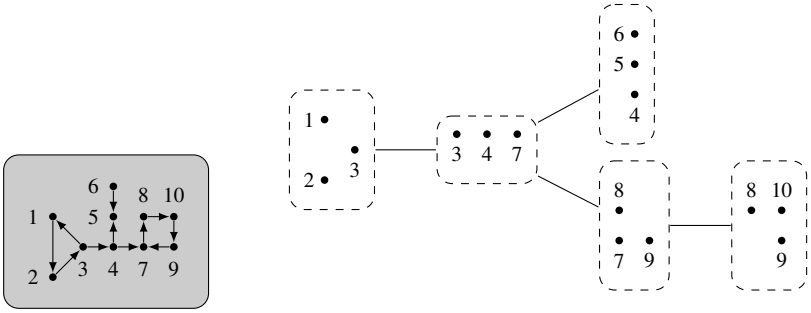
[1] In the following, v, w will range over nodes of hypergraphs, e over edges of hypergraphs, t over nodes of simple graphs and b over edges of simple graphs. In all cases, subscripts may also be used.

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



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

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



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

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

– for each edge $e \in E$, there is a node t of T such that all nodes v connected 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 $wd(\mathcal{T}) = \left(\max_{t \in T} |X_t|\right) - 1$. 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:

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

Example 1 *As examples we consider only unlabeled directed graphs, that is we take* $\Sigma = \{\star\}$ *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 at least* 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 such that trees have treewidth 1. Furthermore discrete graphs have pathwidth and treewidth 0 and an n-clique has both pathwidth and treewidth $n-1$. 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.

3 Cospans as Building Blocks for Graphs

Cospans of graphs can be viewed as operations on graphs with interfaces (in the sense of Courcelle [Cou90, BC87]). Let G be a graph with external nodes (which itself can be represented by a cospan $g\colon \emptyset \to G \leftarrow I$, where the interface I represents the external nodes) and $c\colon I \to H \leftarrow K$ a cospan. By composing g and c we obtain a cospan $(g\,;c)\colon \emptyset \to GH \leftarrow K$, 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 to make the pushout diagram commute. That is, by composing with a cospan we can *add* new nodes, *fuse* existing nodes and *change* the interface of a graph.

There exist finite sets of cospans (called *atomic cospans*) from which, together with disjoint union, all graphs with interfaces can be built; see for example [GH97] and [BK06]. 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: $vertex^n\colon D_n \hookrightarrow D_{n+1}$. This cospan is defined as

  $$vertex^n = D_n \xrightarrow{id'} D_{n+1} \xleftarrow{id} D_{n+1},$$

  where $id'(x) = x$ for $x \leq n$.

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

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

- Add an edge: $connect^n_{A,\theta}\colon D_n \hookrightarrow D_n$, 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

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

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

- Permute the order of the nodes in the interface: $perm^n_\pi\colon D_n \hookrightarrow D_n$. This cospan is defined as

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

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

The atomic cospans are graphically depicted in Figure 3.

Lemma 1 *Let $c = J \to_{c_L} G \leftarrow_{c_R} K$ be a cospan such that J, K are discrete and c_L, c_R are monos. Then there exist atomic cospans $a_1, \ldots, a_n$ such that $c = a_1\,;\cdots;\,a_n$.*

Moreover, the following condition holds for this atomic cospan decomposition: 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 $1 \leq i \leq n$.

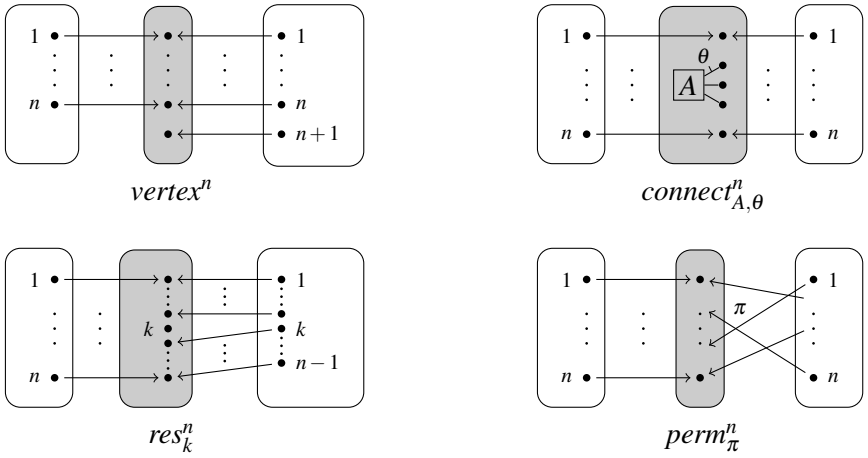


Figure 3: Graphical representations of the atomic cospans.

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 2 Let G be a graph and $\vec{c} = c_1, \ldots, c_n$ 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 3 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 consist 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[3] and it holds for all edges e of G, with $att(e) = v_1 \ldots v_m$, that $v_1, \ldots, v_m$ occur together in some interface.

[3] 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).

(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: Atomic $\subset$ Interface-bag $\subset$ Graph-bag $\subset$ All.

Definition 4 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 5

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

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

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

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

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

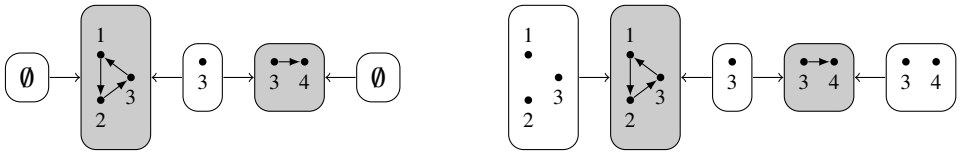
(i) Let $\mathcal{P}$ be a path decomposition of a graph G. There exists a graph-bag decomposition $\vec{c}$ of G such that $wd_{\text{gb}}(\vec{c}) = wd(\mathcal{P})$.

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

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

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

Theorem 1 *For every graph G, $pwd(G) = cpwd_{\text{gb}}(G) = cpwd_{\text{ib}}(G) = cpwd_{\text{at}}(G)$.*



(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

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 3: 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.

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

$$vertex^0 \,;\, vertex^1 \,;\, vertex^2 \,;\, connect^3_{A,\theta^3_{1,2}} \,;\, connect^3_{A,\theta^3_{1,3}} \,;\, connect^3_{A,\theta^3_{2,3}} \,;\, res^3_0 \,;\, res^2_0 \,;$$
$$vertex^1 \,;\, connect^2_{A,\theta^2_{1,2}} \,;\, res^2_0 \,;\, res^1_0,$$

where $\theta^n_{i_1,\ldots,i_m} : \mathbb{N}_m \to \mathbb{N}_n$ denotes the function which maps $\theta(1) = i_1, \ldots, \theta(m) = i_m$.

More details concerning the conversion of the various cospan decompositions into each other can be found in the proof of Lemma 2 in the full version of this paper [BBFK11].

5 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 is a finite set of morphisms with the same domain, while a costar is a finite set of morphisms with the same codomain.

As in the case of (linear) cospan decompositions, 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 5). As in the case of cospan decompositions, we restrict our attention to injective morphisms.

Definition 6

(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 $Colim(\mathcal{C}) = G$.

(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 $Colim(\mathcal{C}) = G$ and additionally 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 $\tau(t)$ 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 7 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 7 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 8 Let G be a graph. The *costar width* ($ctwd_{\mathrm{co}\star}(G)$), *star width* ($ctwd_{\star}(G)$) and *atomic star width* ($ctwd_{\mathrm{at}\star}(F)$) of G are defined as follows:

$$ctwd_{\mathrm{co}\star}(G) = \min\{wd_{\star}(\mathcal{C}) \mid \mathcal{C} \text{ is a costar decomposition of } G\}$$
$$ctwd_{\star}(G) = \min\{wd_{\star}(\mathcal{S}) \mid \mathcal{S} \text{ is a star decomposition of } G\}$$
$$ctwd_{\mathrm{at}\star}(G) = \min\{wd_{\star}(\mathcal{S}) \mid \mathcal{S} \text{ is an atomic star decomposition of } G\}$$

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

Lemma 3

(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 $wd_{\star}(\mathcal{S}') = wd_{\star}(\mathcal{S})$.*

Theorem 2 *For every graph G, $twd(G) = ctwd_{\mathrm{co}\star}(G) = ctwd_{\star}(G) = ctwd_{\mathrm{at}\star}(G)$.*

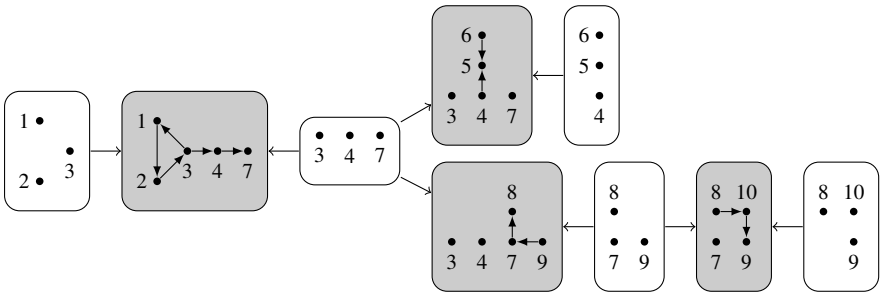
Example 3 *We consider the graph G_T and the tree decomposition $\mathcal{T}$ of Example 1. In order to construct a star decomposition of G_T, we take a cospan for each of the four edges (of the tree) of $\mathcal{T}$. The interfaces of these four cospans are the discrete graphs corresponding to the 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 5a. 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 5b. 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.

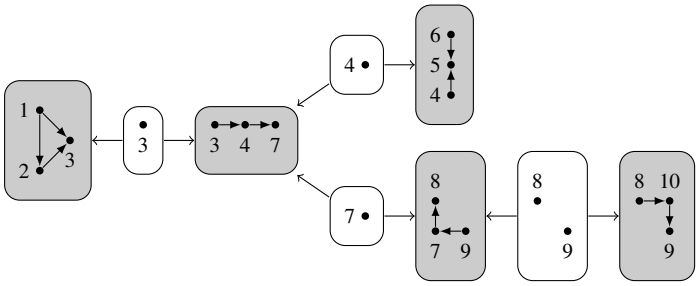
More details concerning the conversion of the various tree and star decompositions into each other can be found in the proof of Lemma 2 in the full version of this paper [BBFK11].

6 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



(a) Star decomposition of G_T



(b) Costar decomposition of G_T

Figure 5: Star and costar decomposition of G_T

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 [Fri10].

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. We are specifically interested in using path and tree decompositions for recognizable graph languages [Cou90] or – more specifically – for graph automata acting as acceptors of graph languages. In order to accept a graph, it is decomposed into atomic units and read – step by step – by the automaton. Depending on whether the state of the automaton, after reading the entire graph, is final, the graph is accepted. In general, decompositions of arbitrary width must be considered, which leads to graph automata with an infinite number of states. For implementation purposes, we have to restrict 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. The size of the state set, however, appears to grow exponentially in the size of the considered maximum width. Therefore we are interested in path and tree decompositions of moderate size, so that graph automata can be implemented efficiently.

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). In the future we also plan to consider automata working on star or costar decompositions by using tree automata [CDG+].

Tree automata have already been used for similar purposes in [CD10, GPW10].

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. [Bod96, BK96]. 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 (see also the investigations in [Küp10]), 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.

Bibliography

[BBFK11] C. Blume, S. Bruggink, M. Friedrich, B. König. Treewidth, Pathwidth and Cospan Decompositions. Technical report, Abteilung für Informatik und angewandte Kognitionswissenschaften, Universität Duisburg-Essen, 2011.

[BBK10] C. Blume, S. Bruggink, B. König. Recognizable Graph Languages for Checking Invariants. In *Proc. of GT-VMC 2010*. 2010.

[BC87] M. Bauderon, B. Courcelle. Graph Expressions and Graph Rewritings. *Mathematical Systems Theory* 20:83–127, 1987.

[BK96] H. L. Bodlaender, T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms* 21(2):358–402, 1996.

[BK06] S. Bozapalidis, A. Kalampakas. Recognizability of graph and pattern languages. *Acta Informatica* 42(8/9):553–581, 2006.

[BK08] S. Bruggink, B. König. On the Recognizability of Arrow and Graph Languages. In *Proc. of ICGT '08*. Springer, 2008. LNCS 5214.

[Bod96] H. L. Bodlaender. A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.* 25(6):1305–1317, 1996.

[CD10] B. Courcelle, I. Durand. Verifying monadic second order graph properties with tree automata. In *European Lisp Symposium*. May 2010.

[CDG+] 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.

[Cou90] B. Courcelle. The Monadic Second-Order Logic of Graphs I. Recognizable Sets of Finite Graphs. *Information and Computation* 85:12–75, 1990.

[Fri10] M. Friedrich. Baumautomaten und Baumzerlegungen für erkennbare Graphsprachen. Master's thesis, Universität Duisburg-Essen, July 2010.

[GH97] F. Gadducci, R. Heckel. An inductive view of graph transformation. In *Proceedings of WADT '97*. Pp. 223–237. 1997.

[GPW10] G. Gottlob, R. Pichler, F. Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Journal of Artificial Intelligence* 174(1):105–132, 2010.

[Hab92] A. Habel. *Hyperedge Replacement: Grammars and Languages*. Springer-Verlag, 1992. LNCS 643.

[HKL93] A. Habel, H.-J. Kreowski, C. Lautemann. A comparison of compatible, finite, and inductive graph properties. *Theoretical Computer Science* 110(1):145–168, 1993.

[Küp10] S. Küpper. Algorithmen für Baum- und Pfadzerlegungen von Graphen. Bachelor's thesis, Universität Duisburg-Essen, October 2010.

[Lau88] C. Lautemann. Decomposition Trees: Structured Graph Representation and Efficient Algorithms. In *Proc. of CAAP '88*. Pp. 28–39. Springer, 1988. LNCS 299.

[Lau91] C. Lautemann. Tree Automata, Tree Decomposition and Hyperedge Replacement. In *Proc. of Graph-Grammars and Their Application to Computer Science '90*. Pp. 520–537. Springer, 1991. LNCS 532.

[LM00] J. J. Leifer, R. Milner. Deriving Bisimulation Congruences for Reactive Systems. In *Proc. of CONCUR 2000*. Pp. 243–258. Springer, 2000. LNCS 1877.

[RS86] N. Robertson, P. Seymour. Graph minors. II. Algorithmic aspects of tree width. *Journal of Algorithms* 7:309–322, 1986.

[SS05] V. Sassone, P. Sobociński. Reactive systems over cospans. In *Proc. of LICS '05*. Pp. 311–320. IEEE, 2005.