Grammars and clique-width bounds from split decompositions

Bruno Courcelle

Grammars and clique-width bounds from split decompositions

Bruno Courcelle
Labri, CNRS and Bordeaux University*
33405 Talence, France
email: courcell@labri.fr

May 22, 2018

Abstract

Graph decompositions are important for algorithmic purposes and for graph structure theory.

We relate the *split decomposition* introduced by Cunnigham to *vertex substitution*, *graph grammars* and *clique-width*.

For this purpose, we extend the usual notion of substitution, upon which *modular decomposition* is based by considering graphs with *dead* vertices. We obtain a simple grammar for distance-hereditary graphs. We also bound the clique-width of a graph in terms of those of the components of a split decomposition that need not be canonical.

For extending these results to directed graphs and their split decompositions (that we handle formally as *graph-labelled trees*), we need another extension of substitution : instead of two types of vertices, dead or *live* as for undirected graphs, we need four types, in order to encode edge directions. We bound linearly the clique-width of a directed graph G in terms of the maximal clique-width of a component arising in a graph-labelled tree that defines G. This result concerns all directed graphs, not only the strongly connected ones considered by Cunningham.

Introduction

Graph decompositions and graph complexity measures such as tree-width and clique-width are important for algorithmic purposes and graph structure theory. They are actually linearly related in many interesting cases [11]. Tree-width,

*This work has been supported by the French National Research Agency (ANR) within the IdEx Bordeaux program "Investments for the future", CPU, ANR-10-IDEX-03-02, and also within the project GraphEn started in October 2015.

clique-width and rank-width (which is equivalent to clique-width with respect to finite boundedness) are defined from hierarchical (tree-based) graph decompositions. Tree-width and clique-width occur as parameters in many *fixed-parameter tractable* (FPT) algorithms [14, 16, 20, 24], in particular for the verification of monadic second-order properties of graphs.

These algorithms are based on the construction of finite automata that run on algebraic terms representing tree-decompositions, or on *clique-width terms* in the case where the parameter is clique-width (see Definition 1.2). However, these automata cannot be implemented in the classical way because their sets of states are much to large. The notion of *fly-automaton* [12, 13] can overcome this difficulty in many cases[1]: these automata compute their transitions instead of looking into transition tables.

Even for checking properties of graphs of bounded tree-width, it is convenient to input graphs by clique-width terms, and we develop the theory and practice of fly-automata for graphs defined in this way (cf. [12, 13, 21]). As for tree-width, computing the exact clique-width of a graph is an NP-complete problem [23]. However, clique-width terms witnessing "good" approximations of clique-width can be used with fly-automata. Such terms can be constructed by different algorithms, and with help of *modular* or *split decomposition*[2], in preliminary steps.

These decompositions are related to clique-width as follows. Each undirected graph has a canonical (unique up to isomorphism) modular decomposition, and also a canonical split decomposition. Its clique-width is the maximal clique-width of a prime module of the modular decomposition (Proposition 2.112 of [14]). Theorems 4.15 and 5.13 prove that it is linearly bounded in terms of the maximal clique-width of a *prime component* of the split decomposition[3], the other components being stars and cliques. These theorems improve the known bounds based of complicated arguments of logic.

Our motivating example is the class of *distance-hereditary graphs* (*DH graphs* in short). They are the undirected graphs G in which the distance in any connected induced subgraph is the same as the one relative to G. They are known to have clique-width at most 3 [27, 34, 35]. However, recognizing the clique-width terms that define them is not easy. For the purpose of testing fly-automata, one may wish to generate large "random" DH graphs, together with the algebraic terms of clique-width 3 that denote them. A good tool consists in using a *context-free graph grammar*, built from clique-width operations that use three labels. The characterization of DH graphs from [1], based on the addition of pendant edges and twins (see Definition 1.1), uses rewriting rules that are not those of a context-free grammar appropriate for using fly-automata intended to run on their derivation trees or on the equivalent clique-width terms. However, from this characterization, we can construct such a context-free grammar based

[1] A software is developped by Irène Durand [21]. Some parts of it are accessible online [22].

[2] See [29] and the references in that article for modular decomposition. Split decomposition defined by Cunnigham [18] is studied in [8, 25]. We give definitions in Sections 4 and 5. Modular and split decomposition can both be computed in linear time for undirected graphs.

[3] Prime components are in some sense undecomposable.

on *vertex-replacement*, a notion developped in [14] (also [7] for an axiomatic approach to context-free graph grammars).

This construction uses a generalization of the standard notion of substitution of a graph for a vertex, that underlies the theory of modular decomposition. We distinguish in a graph H some vertices as *dead* : they will not be linked to any others in case H is substituted into another graph[4]. We use this notion of substitution to define an associated notion of context-free *vertex-replacement grammar* (see [7, 14] for these graph grammars). We obtain a very simple grammar for DH graphs. Distance-hereditary graphs are also easily characterized in terms of their canonical split decompositions, and our grammar somehow translates these decompositions.

Generalizing this observation, we use our grammars to generate the graphs whose prime components (relative to split decomposition) belong (up to isomorphism) to a fixed finite set $\mathcal{M}$. We bound their clique-width in terms of the maximal clique-width of a graph in $\mathcal{M}$, in a better way than what is known from [8].

Split decomposition[5] for strongly connected (directed) graphs has also been studied in [8, 18]. We extend to directed graphs our results for undirected graphs. For expressing split decompositions in terms of graph substitution, we need a more involved notion of substitution. Whereas for undirected graphs, we distinguish dead vertices from *live* ones (the others), for directed graphs, we need three types of live vertices, in order to encode three types of connections between two vertices u and v : from u to v (only), from v to u (only) or in both directions.

We consider split decompositions of directed graphs that need not be strongly connected, and we handle them formally as *graph-labelled trees*, a notion used in [8, 25]. We prove that the clique-width of a directed graph G is bounded by $8k+1$, where k is the maximum clique-width of a component of a graph-labelled tree that defines G, or of its canonical split decomposition when it is strongly connected.

To summarize, the purpose of this article is to clarify the close relationships between split decomposition, clique-width and vertex-replacement graph grammars based on vertex substitutions. In particular, we translate split decompositions of undirected graphs into graph grammars (based on appropriate substitutions to vertices) and we bound linearly the clique-width of a decomposed, directed or undirected graph, in terms of those of the components.

Section 1 is devoted to basic definitions. Section 2 introduces vertex substitutions for undirected graphs with dead vertices, and the corresponding grammars. Section 3 relates clique-width and substitutions. Section 4 studies split decomposition of undirected graphs in this perspective, with the help of graph-labelled

[4] In [34], such vertices are defined in clique-width terms by *inactive* labels. We will use $\perp$ as inactive label in Sections 3 and 5.

[5] A different notion of split decomposition for directed graphs, that generalizes also the one for undirected graphs, has been defined in [32]. We will say a few words about it at the end of Section 5.

trees. Section 5 develops the case of directed graphs.

Acknowledgement: I thank I. Durand, E. Gioan, M. Kanté, S. Oum and C. Paul for their useful comments. I also thank the editors of the special issue of *Discrete Applied Mathematics* relative to the GROW meeting in Toronto, in 2017 for wellcoming this submission.

1 Graphs and clique-width

Most definitions are well-known, we mainly review notation. We state a few facts that are either well-known or easy to prove.

The union of two sets is denoted by $\uplus$ in cases where we stress they are disjoint. The cardinality of a set X is denoted by $|X|$ and its powerset by $\mathcal{P}(X)$. The set of integers $\{1, ..., n\}$ is denoted by $[n]$.

All trees and graphs are nonempty and finite.

Trees

The set of *nodes* of a tree T is denoted by N_T, its set of *leaves*, i.e., of nodes of degree 1, by L_T.

If T has a root, denoted by $root_T$, then $<_T$ denotes the corresponding *ancestor relation*, a strict partial order on N_T (a node is not an ancestor of itself). The root is the unique maximal element and the leaves different from the root are the minimal ones. A *star* S_n is a tree with $n-1 \geq 2$ leaves linked to a single node called its *center*.

Graphs

We consider *simple graphs*, *i.e.*, that are loop-free and without parallel edges. Graphs are directed or not. Directed graphs may have pairs of opposite edges. Undefined notions are as in [19]. A graph G has vertex set V_G and edge set E_G. The corresponding binary adjacency relation is denoted by edg_G. If G is undirected, we denote by uv or vu an edge between u and v. If G is directed, we denote by uv an edge from u to v.

We denote by $G[X]$ the induced subgraph of G with vertex set $X \subseteq V_G$, by $G-X$ the graph $G[V_G - X]$ and by $G-x$ the graph $G[V_G - \{x\}]$ where $x \in V_G$.

Definition 1.1 : *Distance hereditary graph*

An undirected graph G is *distance hereditary* (DH in short) if the distance of two vertices in every connected induced subgraph is the same as in G. For an example, the cycle C_4 (with 4 vertices) is DH whereas C_5 is not. The DH graphs are characterized as follows[6] : a DH graph is an isolated vertex, or the disjoint union of two DH graphs or is obtained from a DH graph by the addition of a pendant edge to a vertex x, or of a *twin* to x. Adding to x a *true twin* is

[6] This characterization is from [1]. The DH graphs are also the graphs of rank-width 1 [35]. They have clique-width at most 3, as we will prove in detail.

adding a new vertex y linked to x and to the neighbours of x. Adding a *false twin* is similar with y not linked to x.

Labelled graphs.

Let C be a finite set of labels. A *C-graph* is a triple $G = (V_G, E_G, \pi_G)$ where π_G is a mapping: $V_G \to C$. Its *type*, denoted by $\tau(G)$ is $\pi_G(V_G)$, *i.e.*, the finite set of labels from C that label some vertex of G.

We denote by $\simeq$ the isomorphism of C-graphs up to vertex labels, *i.e.*, the isomorphism of the underlying unlabelled graphs, and by $\equiv$ the existence of an isomorphism that respects labels. An *abstract C-graph* is an equivalence class of graphs for $\equiv$.

Clique-width

Clique-width is based on operations that modify or combine C-graphs. There will be some restrictions regarding the special label $\perp$ (see Section 3 below).

Definition 1.2 : *Clique-width*

(a) We define the following operations on C-graphs :

- the union of two disjoint C-graphs is denoted by the binary function symbol[7] $\oplus$,
- the unary operation $add_{a,b}$ for $a, b \in C$, $a \neq b$ that adds an undirected edge between each a-labelled vertex x and each b-labelled vertex y (unless there is already an edge xy); for building directed graphs, we use similarly $\overrightarrow{add}_{a,b}$ to add directed edges from a-labelled to b-labelled vertices,
- the unary operation[8] $relab_h$ that changes every vertex label a into $h(a)$ where h is a mapping from C to C,
- and, for each $a \in C$, the nullary symbol $\mathbf{a}(x)$ that denotes the isolated vertex x labelled by a.

We denote by F_C as the set of these operations[9]. A term over F_C is *well-formed* if no two occurrences of nullary symbols designate the same vertex ; in particular, the graphs defined by the two arguments of an operation $\oplus$ are disjoint. We denote by $T(F_C)$ the set of well-formed terms. We call them *clique-width terms*. Each such term t denotes a C-graph $val(t)$ whose vertices are exactly those specified by the nullary symbols of t. Its *width* is the number of labels that occur in t.

[7] As $\oplus$ is associative, we will write $t = t_1 \oplus t_2 \oplus ... \oplus t_n$ instead of $t_1 \oplus (t_2 \oplus (... \oplus t_n)...)$.

[8] If h only changes a into b, we denote $relab_h$ by $relab_{a \to b}$ and call this operation an *elementary relabelling*. By using only elementary relabellings, we obtain the same notion of clique-width ([14], Proposition 2.118).

[9] Vertices are taken from a fixed countable set.

The *clique-width* of a labelled graph[10] G, denoted by $cwd(G)$, is the least width of a term t such that $G \simeq val(t)$. We denote by $cwd^*(G)$ the least width of t such that $G \equiv val(t)$. Hence, $cwd(G) \leq cwd^*(G)$. Clearly, $cwd(G) = cwd^*(G')$ where G' is obtained from G by relabelling all its vertices in the same way.

Here are some examples. The clique-width of a tree is at most 3, that of the clique K_n is 2 for $n \geq 2$. The undirected cycles C_3, C_4 have clique-width 2, C_5, C_6 have clique-width 3, and C_n has clique-width 4 for $n \geq 7$. For directed cycles $\overrightarrow{C}_n$, we have $cwd(\overrightarrow{C}_3) = 3$ and $cwd(\overrightarrow{C}_n) = 4$ if $n \geq 4$.

Lemma 1.3 : For every labelled graph G, we have :
$$\max\{|\tau(G)|, cwd(G)\} \leq cwd^*(G) \leq |\tau(G)| \cdot cwd(G).$$

Proof: The first inequality is clear from definitions. To prove the second one, we assume without loss of generality, that the type of G is $[p]$. Let H be G with all vertices labelled in the same way. Let C be the set of k labels of a term t that defines H. For each a in C and $i \in [p]$, we define a new label (a, i) that will only label the vertices x such that $\pi_G(x) = i$.

Consider in t a nullary symbol $\boldsymbol{a}(x)$. If $\pi_G(x) = i$, we replace it by $(\boldsymbol{a}, \boldsymbol{i})(x)$.

Each relabelling $relab_h$ is replaced by $relab_{h'}$ where h' maps (a, i) to (b, i) whenever h maps a to b. Similarly, we replace $add_{a,b}$ by the composition of the operations $add_{(a,i),(b,j)}$ for $i, j \in [p]$. We obtain in this way a term[11] t'. We let $h'' : C \times [p]$ map each label (a, i) to i for each $a \in C$.

Then $G = relab_{h''}(val(t')) = val(relab_{h''}(t'))$. The term $relab_{h''}(t')$ uses at most $p + p.k$ labels. However, we can fix some $a \in C$ and replace everywhere (a, i) by i, for each i. We obtain a term of width at most $p.k$ that defines G. □

Hence if the type of G consists of p labels and k is the clique-width of the corresponding unlabelled graph, then one can define G, with its labelling, by a term with at most $p.k$ labels. This lemma implies that $cwd(G) \leq 2cwd(G - x)$ if x is a vertex of G. This bound is proved in [28]. Can it be improved[12] ? Can one improve the bound $p.k$ in Lemma 1.3[13] ?

We will also use nullary symbols $\mathbf{a}$ that do not designate any particular vertex. In this case, the vertex defined by an occurrence u of $\mathbf{a}$ in a term is u itself. We will also consider that a term written with such nullary symbols denotes an abstract graph. See [14], Section 2.52. We will denote by $\overline{F}_C$ the signature F_C where the symbols $\mathbf{a}(x)$ are replaced by $\mathbf{a}$, and by $\overline{t}$ the term in $T(\overline{F}_C)$ obtained from a term $t \in T(F_C)$ by replacing each by $\mathbf{a}(x)$ by $\mathbf{a}$. Then $val(\overline{t}) \equiv val(t)$.

[10] In [14], we denote cwd^* by cwd.

[11] The construction is similar for directed graphs and it needs no more labels.

[12] It not hard to prove that $lcwd(G) \leq lcwd(G - x) + 2$ where $lcwd$ denotes the *linear clique-width*. This variant is defined by requiring that at least one of the two arguments of an operation $\oplus$ is a nullary symbol. See e.g., [14, 30].

[13] A lower-bound is established in [17]. .

2 Substitution to vertices

We consider undirected graphs in this section. We will adapt the definitions for directed graphs in Section 5.

Let C be a set of labels containing $\perp$. The vertices of a graph G labelled by $\perp$ will be said to be *dead* ; they form the set V_G^{dead}. The others, said to be *live,* form the set V_G^{live}. The unary operation κ, read *kill*, relabels all vertices by $\perp$ hence makes them dead.

Definition 2.1 : *Substitution.*

Let K be a C-graph and $x_1, \ldots, x_p$ be pairwise distinct vertices. Let $H_1, \ldots, H_p$ be pairwise disjoint C-graphs, that are disjoint[14] from K. We define a C-graph as follows:

$G := K[H_1/x_1, \ldots, H_p/x_p]$ where
$V_G := (V_K - \{x_1, \ldots, x_p\}) \uplus V_{H_1} \uplus \ldots \uplus V_{H_p}$,
$\pi_G(v) := \pi_K(v)$ if $v \in V_K - \{x_1, \ldots, x_p\}$,
$\pi_G(v) := \pi_K(x_i)$ if $v \in V_{H_i}^{live}$,
$\pi_G(v) := \perp$ if $v \in V_{H_i}^{dead}$.

Its edges are as follows, for u, v in V_G :

$uv \in E_G$ if and only if :
either $uv \in E_K$ and neither u nor v is in $\{x_1, \ldots, x_p\}$,
or $uv \in E_{H_i}$ for some i,
or $u \in V_K$, $ux_i \in E_K$ and $v \in V_{H_i}^{live}$ (or vice-versa since we define undirected graphs),
or $u \in V_{H_i}^{live}$, $v \in V_{H_j}^{live}$ and $x_i x_j \in E_K$ (so that $i \neq j$).

The type of G is thus that of K, possibly augmented with $\perp$ if some H_i has dead vertices (these vertices are dead in G). The labels of K have no influence on the definition of the edges of G. They are only used for defining the labels of the resulting graph G. The labels of the graphs H_i other than $\perp$ do not appear in G: hence, if we have $h_i(a) = \perp$ if and only if $a = \perp$, for each i, then :

$$K[H_1/x_1, \ldots, H_p/x_p] = K[relab_{h_1}(H_1)/x_1, \ldots, relab_{h_p}(H_p)/x_p].$$

If all vertices of H_p are dead, then

$$K[H_1/x_1, \ldots, H_p/x_p] = (K - x_p)[H_1/x_1, \ldots, H_{p-1}/x_{p-1}] \oplus H_p.$$

[14] It is actually enough to assume that $(V_{H_1} \uplus \ldots \uplus V_{H_p}) \cap (V_K - \{x_1, \ldots, x_p\}) = \emptyset$.

Because of dead vertices, this notion of substitution differs from the classical one, use in particular in the theory of modular decomposition (see the survey [29]).

Proposition 2.2 : Let K, H_1, H_2 be pairwise disjoint C-graphs and $x_1 \in V_K$.

(1) If x_2 is another vertex of K, then $K[H_1/x_1][H_2/x_2] = K[H_1/x_1, H_2/x_2]$.

(2) If $x_2 \in V_{H_1}$, then $K[H_1/x_1][H_2/x_2] = K[H_1[H_2/x_2]/x_1]$.

Proof : (1) Straightforward verification from the definitions.

(2) Let $G := K[H_1/x_1][H_2/x_2]$ and $G' := K[H_1[H_2/x_2]/x_1]$.

Clearly, $V_G = V_{G'}$.

Let u, v belong to V_G. If u and v are both, either in V_K, or in V_{H_1} or in V_{H_2}, then $uv \in E_G$ if and only if $uv \in E_{G'}$. Otherwise we distinguish three cases.

(i) $u \in V_K$ and $v \in V_{H_1}$; then, $uv \in E_G$ if and only if $ux_1 \in E_K$ and v is live in H_1, if and only if $uv \in E_{G'}$.

(ii) $u \in V_K$ and $v \in V_{H_2}$; then, $uv \in E_G$ if and only if $ux_2 \in E_{K[H_1/x_1]}$ and v is live in H_2. The condition $ux_2 \in E_{K[H_1/x_1]}$ is equivalent to : $ux_1 \in E_K$ and x_2 is live in H_1.

Now $uv \in E_{G'}$ if and only if $ux_1 \in E_K$ and v is live in $H_1[H_2/x_2]$ which is true if and only if v is live in H_2 and x_2 is live in H_1. Hence, $uv \in E_G$ if and only if $uv \in E_{G'}$.

(iii) $u \in V_{H_1} - \{x_2\}$ and $v \in V_{H_2}$; then $uv \in E_G$ if and only if $ux_2 \in E_{H_1}$ and v is live in H_2, if and only if $uv \in E_{H_1[H_2/x_2]}$, if and only if $uv \in E_{G'}$.

It remains to verify that $\pi_G = \pi_{G'}$.

If $u \in (V_K - \{x_1\}) \uplus (V_{H_1} - \{x_2\})$, then $\pi_G(u) = \pi_{G'}(u)$ because u is not affected by the substitutions to x_2.

If $u \in V_{H_2}$, then $\pi_G(u) = \perp$ if u is dead in H_2; it is $\pi_{K[H_1/x_1]}(x_2)$ otherwise; but $\pi_{K[H_1/x_1]}(x_2) = \perp$ if x_2 is dead in H_1, otherwise $\pi_{K[H_1/x_1]}(x_2) = \pi_K(x_1)$. Now, $\pi_{G'}(u) = \perp$ if u is dead in $H_1[H_2/x_2]$ and it is $\pi_K(x_1)$ otherwise; observe that u is dead in $H_1[H_2/x_2]$ if and only if it is dead in H_2 or x_2 is dead in H_1. We obtain $\pi_G(u) = \pi_{G'}(u)$ in this case; this value is either $\pi_K(x_1)$ or $\perp$ (if u is dead in H_2 or x_2 is dead in H_1).

This completes the proof. □

In [7], Properties (1) and (2) are called respectively *commutativity* and *associativity of substitution*. They are axioms for the definition of *context-free graph grammars* based on an abstract notion of substitution. These grammars are particular vertex replacement grammars [14].

Definition 2.3 : *Graph operations based on substitution.*

(a) For each C-graph K with vertex set enumerated as $\{x_1, \ldots, x_p\}$, we define as follows a p-ary graph operation[15] on C-graphs denoted by $\sigma[K, x_1, \ldots, x_p]$:

$$\sigma[K, x_1, \ldots, x_p](H_1, \ldots, H_p) := K[H_1/x_1, \ldots, H_p/x_p]$$

[15] We use the notation $\sigma[K, v_1, \ldots, v_p]$ for the function symbol and the corresponding operation.

where $H_1, \ldots, H_p$ are pairwise disjoint C-graphs that are disjoint from K. Note that the vertex set of $\sigma[K, v_1, \ldots, v_p](H_1, \ldots, H_p)$ is $V_{H_1} \uplus \ldots \uplus V_{H_p}$.

If $H_1, \ldots, H_p$ are not pairwise disjoint, we replace them by isomorphic copies in a standard way ([14], Chapter 2), so that $\sigma[K, x_1, \ldots, x_p]$ becomes a p-ary operation on abstract C-graphs, (*i.e.*, isomorphism classes of C-graphs).

We denote by Σ_C the countable set of these operations together with the nullary symbols $\boldsymbol{a}(x)$ to denote vertices.

A term in $T(\Sigma_C)$ is *well-formed* if each vertex x occurs at most once in a symbol $\boldsymbol{a}(x)$. Every well-formed term t denotes a graph $val(t)$. The signature $\overline{\Sigma}_C$ is obtained from Σ_C by replacing, for each a, the different symbols $\boldsymbol{a}(x)$ by $\boldsymbol{a}$. As for clique-width terms in $T(\overline{F}_C)$, each term in $T(\overline{\Sigma})$ denotes an abstract graph.

We denote by $relab_a$ the relabelling that replaces by a every label except $\perp$.

Proposition 2.4 : Let $t, t' \in T(\Sigma_C)$ and x be a vertex in $val(t)$ defined by a nullary symbol $\boldsymbol{a}(x)$. Let t' be a term such that $V_{val(t')} \cap (V_{val(t)} - \{x\}) = \emptyset$. Then, we have :

$$val(t)[val(t')/x] = val(t[relab_a(t')/\boldsymbol{a}(x)]).$$

The term $t[relab_a(t')/\boldsymbol{a}(x)]$ is obtained by substituting t' in t for the unique occurrence of $\boldsymbol{a}(x)$. It is well-defined because $V_{val(t')} \cap (V_{val(t)} - \{x\}) = \emptyset$ (cf. the footnote in Definition 2.1).

Proof : By induction of the structure of t.

If $t = \boldsymbol{a}(x)$, then $val(t[relab_a(t')/\boldsymbol{a}(x)] = val(relab_a(t')) = \boldsymbol{a}(x)[val(t')/x])$ by the definition of substitution, hence is equal to $val(t)[val(t')/x]$.

Let $t = \sigma[K, v_1, \ldots, v_p](t_1, \ldots, t_p)$. Without loss of generality and to simplify notation, we assume that $\boldsymbol{a}(x)$ occurs in t_1. Then,

$$t[t'/\boldsymbol{a}(x)] = \sigma[K, v_1, \ldots, v_p](t_1[t'/\boldsymbol{a}(x)], t_2, \ldots, t_p);$$
$$val(t[t'/\boldsymbol{a}(x)]) \ = K[val(t_1[t'/\boldsymbol{a}(x)])/v_1, val(t_2)/v_2, \ldots, val(t_p)/v_p].$$

By induction :

$$val(t_1[t'/\boldsymbol{a}(x)]) = val(t_1)[val(t')/x],$$

hence

$$val(t[t'/\boldsymbol{a}(x)]) \ = K[val(t_1)[val(t')/x]/v_1, \ldots, val(t_p)/v_p]$$
$$= K[val(t_1)/v_1, \ldots, val(t_p)/v_p][val(t')/x] \text{ by Propoposition 2.2(2),}$$
$$= val(t)[val(t')/x]. \qquad \square$$

From these operations one can define *grammars*, formalized by *systems of recursive equations* in sets of abstract C-graphs of which one takes least solutions (cf. [14], Chapter 3 for "context-free" grammars in a universal algebra setting). We first consider some examples.

Definitions 2.5 : *Some basic operations.*
Here are operations of particular interest. They concern D-graphs where $D := \{\bot, \top\}$ (and $\top$ labels live vertices).

$H_1 \oplus H_2 = \sigma[K, x_1, x_2](H_1, H_2)$ where K consists of two isolated live vertices x_1 and x_2.

$H_1 \otimes H_2 := \sigma[K, x_1, x_2](H_1, H_2)$ where K is the edge x_1x_2 and x_1 and x_2 are live.

We will also use :

$\Lambda(H_1, H_2) := \sigma[K, x_1, x_2](H_1, H_2)$ where K is the edge x_1x_2, x_1 is live and x_2 is dead.

We have :

$\kappa(H) = \sigma[K, x_1](H)$ where K consists of the dead vertex x_1.

The operations $\oplus$ and $\otimes$ are associative and commutative. Here are some algebraic properties of the other ones:

$$\Lambda(\Lambda(G, H_1), H_2) = \Lambda(\Lambda(G, H_2), H_1) = \Lambda(G, H_1 \oplus H_2),$$
$$\kappa(\Lambda(G, H)) = \kappa(G \otimes H),$$
$$\Lambda(\kappa(G), H) = \kappa(G) \oplus \kappa(H).$$

We let $\overline{\Sigma}_{dh}$ be the signature $\{\oplus, \otimes, \Lambda, \kappa, \top\} \subseteq \overline{\Sigma}_D$. We need not use the nullary symbols $\bot(x)$ because a dead vertex x can be defined by $\kappa(\top(x))$. Actually, a vertex introduced as dead remains isolated.

Examples 2.6 : *Some grammars (equations in sets of D-graphs) over* $\overline{\Sigma}_{dh}$.
(1) The equation

$$X = \top \cup (X \oplus X) \cup (X \otimes X)$$

defines the set of cographs (all vertices are live, as $\oplus$ and $\otimes$ do not introduce dead vertices). Here X denotes a set of abstract $\{\top\}$-graphs, $X \oplus X$ denotes $\{G \oplus H \mid G, H \in X\}$, and similarly for the other operations.

This equation can also be solved in the set of terms $T(\{\oplus, \otimes, \top\})$. Then X denotes a subset of $T(\{\oplus, \otimes, \top\})$ and $X \oplus X$ denotes the set of terms $\{t \oplus t' \mid t, t' \in X\}$ (and similarly for $\otimes$). We obtain the set of terms $L(X)$.

More generally, for a system of recursive equations with unknowns X, Y, Z, ... we denote by $L(X), L(Y), L(Z)$,... the sets of terms. A fundamental property of these systems ([14], Proposition 3.23) establishes that the corresponding sets of objects X, Y, Z (elements of the relevant algebra) are the sets of *values of the terms* in $L(X), L(Y), L(Z)$,... Here, we get that the cographs are the values of the terms in $L(X)$, where terms are evaluated as abstract graphs. In the present case, we have $L(X) = T(\{\oplus, \otimes, \top\})$, hence the cographs are defined by all terms in $T(\{\top, \oplus, \otimes\})$.

(2) The rooted trees are defined by the equation

$$R = \top \cup \Lambda(R, R).$$

In a tree defined in this way, only the root is live. Another grammar for trees, using two equations, is :

$$Y = \top \cup \Lambda(\top, Z),$$
$$Z = Y \cup (Z \oplus Z).$$

Here, Z defines *forests*, i.e., nonempty disjoint unions of rooted trees. Trees (without distinguished root) are defined by the additional equations $T = \kappa(R)$ or $T = \kappa(Y)$.

For example, the tree with nodes u, v, w, x, y, z, root x and edges xy, xz, xv, zu and vw is defined by the term :

$$\Lambda(\Lambda(\Lambda(\top(x), \top(y)), \Lambda(\top(z), \top(u))), \Lambda(\top(v), \top(w)))$$

belonging to $L(Y)$ or by the term of $L(U)$:

$$\Lambda(\top(x), \top(y) \oplus \Lambda(\top(z), \top(u)) \oplus \Lambda(\top(v), \top(w))).$$

The paths with one live vertex at one end are defined by the equation

$$P = \top \cup \Lambda(\top, P).$$

(3) We will prove below that the equation
$W = \top \cup (W \oplus W) \cup (W \otimes W) \cup \Lambda(W, W)$
defines, up to vertex labels, the distance hereditary graphs.

From these equations, we obtain that the rooted trees are defined by *all terms* in $T(\{\Lambda, \top\})$ or by *certain terms* in $T(\{\oplus, \Lambda, \top\})$, and that the distance hereditary graphs are defined by terms in $T(\{\oplus, \otimes, \Lambda, \top\})$. In these equations and the generated terms, the label $\perp$ for dead vertices does not appear explicily, but it is introduced by the operations Λ and κ. $\square$

In the following description of DH graphs, all vertices are defined as dead, equivalently, unlabelled (there is no reason to distinguish any vertices as "roots").

The following recursive definition of DH graphs has been established in [5], but we think interesting to prove it by using the concepts of this article. We recall that equation systems always define abstract graphs.

Proposition 2.7 : (1) The DH graphs form the set X defined by the two equations :

$$X = \kappa(W),$$
$$W = \top \cup (W \oplus W) \cup (W \otimes W) \cup \Lambda(W, W).$$

(2) The connected DH graphs form the set Y defined by the equation :

$$Y = \kappa(\top) \cup \kappa(W \otimes W)$$

and the equation of (1) that defines W.

Proof : (1) Note that[16] $L(W) = T(\{\oplus, \otimes, \Lambda, \top\})$. For both direction we will use the characterization of DH graphs recalled in Definition 1.1.

(a) Every DH graph G is in the set $val(X) = val(\kappa(W))$.

We use induction on the number n of vertices of G.

If $n = 1$, then G is a single dead vertex, hence is $G \equiv \kappa(\top) \in \kappa(W)$.

Otherwise there are four cases.

(i) G is the disjoint union of two DH graphs H, H'. Then, $H \equiv \kappa(t), H' \equiv \kappa(t')$ for some $t, t' \in W = T(\{\oplus, \otimes, \Lambda, \top\})$. Hence, $G \equiv \kappa(t \oplus t')$ where $t \oplus t' \in W$ because $\kappa(t \oplus t') \equiv \kappa(t) \oplus \kappa(t')$.

(ii) If G is obtained from a DH G' by adding a pendant vertex y to a vertex x of G', we have $G = G'[H/x]$ where H is the edge xy, with x live and y dead[17]; hence $H = val(\Lambda(\top(x), \top(y)))$.

We have $G' = \kappa(val(t'))$ where t' is a well-formed term over $\oplus, \otimes, \Lambda$ and the nullaries that define vertices. It has one occurrence of $\top(x)$.

We let $t := t'[\Lambda(\top(x), \top(y))/\top(x)]$. By Proposition 2.4, we have $val(t) = val(t')[H/\top(x)]$. Hence $G = \kappa(t)$, so that $G \equiv \kappa(\overline{t})$ where $\overline{t} \in W$ is obtained from t by replacing by $\top$ the symbols $\top(z)$ that define vertices.

(iii) If G is obtained from a DH G' by adding a false twin y to a vertex x. We have $G = G'[H/x]$ where H consists of two isolated live vertices x and y. Hence $H = val(\top(x) \oplus \top(y))$. The proof continues as in (ii) with $\top(x) \oplus \top(y)$ instead of $\Lambda(\top(x), \top(y))$.

(iv) If G is obtained from DH G' by adding a true twin y to a vertex x. Here $G = G'[H/x]$ where H consists of two live vertices x and y linked by an edge, hence $H = val(\top(x) \otimes \top(y))$. The proof continues as in (iii) with $\top(x) \otimes \top(y)$ instead of $\top(x) \oplus \top(y)$.

(b) Conversely, let $G = val(t)$ for some well-formed term t over $\oplus, \otimes, \Lambda$ and the nullaries that define vertices. We prove that $\kappa(G)$ is DH by induction on the size of t.

[16] See Example 2.5(1) for the solution of equations in sets of terms. In the proof, we will identify W and $L(W)$ without risk of difficulty.

[17] We use here the footnote in Definition 2.1.

If $t = \top(x)$, the result holds because an isolated vertex is DH. Otherwise, we can find a position u in t such that t/u, the subterm of t issued from position u is either $\Lambda(\top(x), \top(y))$ or $\top(x) \oplus \top(y)$, or $\top(x) \otimes \top(y)$. Then $t = t'[(t/u)\ /\top(x)]$ for some well-formed term t' (obtained by replacing in t the subterm t/u by $\top(x)$). By induction, $\kappa(val(t'))$ is a DH graph G' and $G = G'[H/x]$ by Proposition 2.6, where H is respectively as in cases (ii), (iii) or (iv).

(2) It is clear that a term t in $L(X)$ defines a connected graph if and only if it is not of the form $\kappa(t_1 \oplus t_2)$. Hence, the connected DH graphs can be defined by the equation :

$$Y = \kappa(\top) \cup \kappa(W \otimes W) \cup \kappa(\Lambda(W, W))$$

where W is as in (1). However, we observed that $\kappa(\Lambda(G, H)) = \kappa(G \otimes H)$ for all D-graphs G and H. Hence, the term $\kappa(\Lambda(W, W))$ can be removed. □

The *bipartite DH graphs* are built from isolated vertices by the addition of pendant edges and of false twins [1]. Hence, they form the set B defined by the two equations $B = \kappa(W')$ and $W' = \top \cup (W' \oplus W') \cup \Lambda(W', W')$.

3 Clique-width and substitution operations.

Substitutions as derived clique-width operations.

We recall[18] that a *derived operation* relative to an algebra $\mathbb{M}$ over a functional signature F is defined by a term t in $T(F, \{u_1, ..., u_p\})$, *i.e.*, a term over F with variables (nullary symbols to which values or terms can be substituted) $u_1, ..., u_p$. The corresponding p-ary function $t_{\mathbb{M}}$ is defined by evaluating t with p arguments from the domain of $\mathbb{M}$ as values of $u_1, ..., u_p$.

For an example using clique-width operations, the operation $\otimes$ on graphs of type $\{\top\}$ (cf. Definition 2.3) adds to the disjoint union of the two argument graphs G and H all possible edges between the vertices of G and those of H. We have $G \otimes H = relab_{a \to \top}(add_{\top,a}(G \oplus relab_{\top \to a}(H))$. Hence, $\otimes$ is defined by the term[19] $relab_{a \to \top}(add_{\bot,a}(u_1 \oplus relab_{\top \to a}(u_2))$.

We let $Lin(F_C, \{u_1, ..., u_p\})$ be the set of terms where each variable u_i has a unique occurrence and no other nullary symbol occurs. Every such term defines a p-ary mapping on C-graphs denoted by $t_{\mathbb{G}}$. For pairwise disjoint graphs $H_1, \ldots, H_p$, the vertex set of $t_{\mathbb{G}}(H_1, \ldots, H_p)$ is $V_{H_1} \uplus \ldots \uplus V_{H_p}$. Our objective is to express the operations $\sigma[K, x_1, \ldots, x_p]$ as derived operations over F_C, the signature upon which clique-width is based.

[18] from [14], Section 2.1

[19] This term uses an auxiliary label a to define graphs of type $\{\top\}$. This label does not appear in the type. It can be replaced by any other one different from $\top$.

We define $T_\perp(F_C)$ as the set of terms that use neither the operations[20] $add_{a,\perp}$, $add_{\perp,b}$ (nor the similar ones that define directed edges), nor $relab_h$ such that $h(\perp) \neq \perp$, nor the nullary symbols $\perp(x)$. We denote by $cwd^\perp(G)$ the minimal cardinality of $C - \{\perp\}$ such that $G \simeq val(t)$ for some term $t \in T_\perp(F_C)$. Clearly, $cwd(G) \leq cwd^\perp(G)+1$. We have $cwd(T) = 3$ and $cwd^\perp(T) = 2$ for any tree T that is not a star.

Let K be a C-graph with vertex set $\{x_1, \ldots, x_p\}$ defined by a term t in $T_\perp(F_C)$. Each vertex x_i is defined by a nullary symbol $\boldsymbol{a_i}(x_i)$ in t ($a_i \neq \perp$). We define $\widehat{t} := t[u_1/\boldsymbol{a_1}(x_1), \ldots, u_p/\boldsymbol{a_p}(x_p)]$. This term is in $Lin(F_C, \{u_1, ..., u_p\})$. We recall that $relab_a$ is the relabelling that replaces by a every label except $\perp$.

Lemma 3.1 : For pairwise disjoint C-graphs $H_1, \ldots, H_p$, we have:

$$\sigma[K, x_1, \ldots, x_p](H_1, \ldots, H_p) = \widehat{t}_{\mathbb{G}}(relab_{a_1}(H_1), \ldots, relab_{a_p}(H_p)).$$

Proof: By induction on the structure of t.

If $t = \boldsymbol{a_1}(x_1)$, then $\widehat{t} = u_1$ and $\sigma[K, x_1](H_1) = relab_{a_1}(H_1)$ by the definition of substitution, hence is $\widehat{t}_{\mathbb{G}}(relab_{a_1}(H_1))$.

If $t = t_1 \oplus t_2$, then, without loss of generality, we assume that the vertices of $K_1 := val(t_1)$ are $x_1, \ldots, x_i$ and those of $K_2 := val(t_2)$ are $x_{i+1}, \ldots, x_p$. We have $\widehat{t} = \widehat{t_1} \oplus \widehat{t_2}$. Then, since substitution distributes over disjoint union[21] and by induction :

$$\begin{aligned}
&\sigma[K, x_1, \ldots, x_p](H_1, \ldots, H_p) = \\
&\sigma[K_1, x_1, \ldots, x_i](H_1, \ldots, H_i) \oplus \sigma[K_2, x_{i+1}, \ldots, x_p](H_{i+1}, \ldots, H_p) = \\
&\widehat{t_1}_{\mathbb{G}}(relab_{a_1}(H_1), \ldots, relab_{a_i}(H_i)) \oplus \widehat{t_2}_{\mathbb{G}}(relab_{a_i}(H_i), \ldots, relab_{a_p}(H_p)) = \\
&\widehat{t}_{\mathbb{G}}(relab_{a_1}(H_1), \ldots, relab_{a_p}(H_p)).
\end{aligned}$$

If $t = f(t_1)$ where f is $relab_h$ or $add_{a,b}$, then the result holds because, for every C-graph K with vertices $x_1, \ldots, x_p$, we have :

$$\sigma[f(K), x_1, \ldots, x_p](H_1, \ldots, H_p) = f(\sigma[K, x_1, \ldots, x_p](H_1, \ldots, H_p)).$$

The equality to be proved follows then by induction. □

We will denote by t_K and $\widehat{t_K}$ terms associated with K as above. We say that an operation $\sigma[K, x_1, \ldots, x_p]$ has *width* k if $cwd^\perp(K) = k$. The operations $\oplus, \otimes, \Lambda$ and κ have respective widths 2,2,2 and 1.

Proposition 3.2 : If $G \equiv val(s)$ for some term s in $T(\Sigma_C)$ whose operations have width at most k, then $cwd^\perp(G) \leq k$ and $cwd(G) \leq k+1$.

Proof : By induction on the structure of s, we define a term $\widetilde{s}$ in $T(F_C)$ such that $val(\widetilde{s}) \equiv val(s)$.

If $s = \boldsymbol{a}(x_1)$, then $\widetilde{s} := s$.

If $s = \sigma[K, x_1, \ldots, x_p](s_1, \ldots, s_p)$ then, we let $\widehat{t_K} \in Lin(F_C, \{u_1, ..., u_p\})$ be a term associated with K as in Lemma 3.1, and we define

[20] These limitations on the use of $\perp$ make it an *inactive label* in [34].

[21] This is clear from Definition 2.1.

$\widetilde{s} := \widehat{t_K}[relab_{a_1}(\widetilde{s_1})/u_1, \ldots, relab_{a_p}(\widetilde{s_p})/u_p]$.

It is clear that $val(\widetilde{s}) \equiv val(s)$.

The set of labels used in $\widetilde{s}$ is the set of all those used in the terms $\widehat{t_K}$. We now bound $cwd^{\perp}(G)$. Without loss of generality, we can assume that all labels of the terms that define the operations that occur in s are in a set C such that $C - \{\perp\}$ has cardinality k. Hence $cwd^{\perp}(G) \leq k$ and $cwd(G) \leq k+1$. $\square$

If in s, all the operations of maximal width k do not use $\perp$ in their definitions by terms, then $cwd(G) = cwd^{\perp}(G) \leq k$.

We obtain that distance hereditary graphs have clique-width at most 3 (known from [27, 35]) because they are defined with the operations $\kappa, \Lambda, \oplus$ and $\otimes$ of width at most 2. As the operation Λ of width 2 needs $\perp$ in its defining term, we do not have $cwd(G) = cwd^{\perp}(G) \leq 2$. Proposition 4.9 of [34] establishes that conversely, G is distance hereditary if $cwd^{\perp}(G) \leq 2$.

4 Split decomposition

The *split decomposition* of directed and undirected graphs has been defined and studied by Cunnigham in [18]. We will formulate it in terms of graph-labelled trees as in [25] (called and *split-decomposition graphs* in [8]). We only consider undirected graphs in this section.

Definition 4.1 : *Graph-labelled trees and the graphs they describe.*

We denote by L_T the set of leaves of a tree T and by $Inc_T(v)$ the set of edges incident to a node v.

(a) A *graph-labelled tree*, denoted by $\mathcal{T}$ is a tree T with at least three nodes that is equipped, for each node $v \in N_T$, with a connected graph H_v, called a *component*, and a bijection $\rho_v : Inc_T(v) \to V_{H_v}$. The components are pairwise disjoint. We identify u and the unique vertex of H_u if u is a leaf.

(b) The corresponding *split-graph* $S(\mathcal{T})$ is the union of the components together with the edges $\rho_u(e)\rho_v(e)$ for $e = uv$, (cf. Figure 1). A path in $S(\mathcal{T})$ is *alternating* if no two consecutive edges are in a same component. Between any two vertices x, y of $S(\mathcal{T})$, there is at most one alternating path. If there is one, we say that x is *accessible from* y. We add *through* z (or e) to indicate that this path goes through a particular vertex z (or edge e). For a vertex w of $S(\mathcal{T})$ belonging to a component H_u, we denote by $A(w)$ (respectively by $P(w)$) the set of vertices accessible from w by a nonempty alternating path whose first edge is not in H_u (reachable from w by a nonempty path in $S(\mathcal{T})$ whose first edge is not in H_u).

(c) The *graph described by* $\mathcal{T}$, denoted by $G(\mathcal{T})$, has vertex set L_T and an edge uv if and only if u is accessible from v. It is connected ([25], Lemma 2.3)

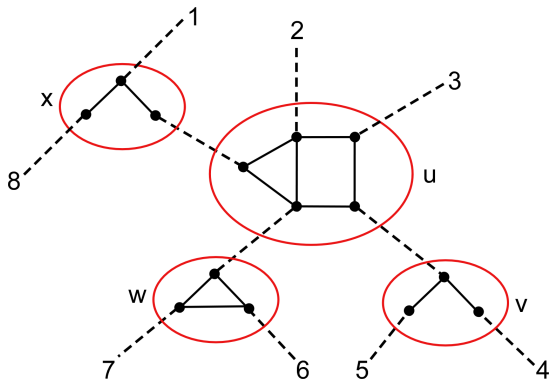

Figure 1: A graph-labelled tree $\mathcal{T}$.

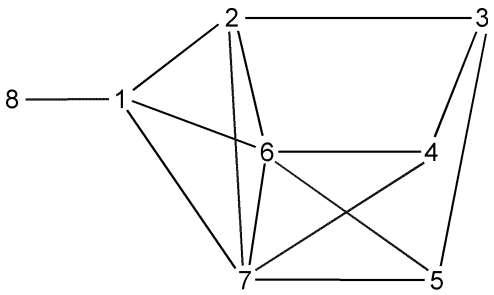

Figure 2: The graph $G(\mathcal{T})$.

because the components are defined as connected[22].

(d) Let $e = uv$ be an edge of T between two internal nodes. The *node-joining operation* (cf. [25]) contracts this edge, hence fuses u and v into a single node say w, giving tree T'; it replaces $H_u \uplus H_v$ by the connected graph $H'_w := H_u \uplus H_v - \{\rho_u(e), \rho_v(e)\}$ and with an edge between any vertex x in H_u and any vertex y in H_v such that $x\rho_u(e) \in E_{H_u}$ and $\rho_v(e)y \in E_{H_v}$. We obtain a graph-labelled tree $\mathcal{T}'$ that describes the same graph. If $\rho_u(e)$ has degree r in H_u and $\rho_v(e)$ has degree s in H_s the resulting component H'_w has a subgraph isomorphic to the complete bipartite graph $K_{r,s}$. The opposite transformation is called *node-splitting*. It preserves also the defined graph. □

Example 4.2 : Figure 1 shows a graph-labelled tree with leaves $1, ..., 8$ and internal nodes u, v, w, x. The associated graph G is in Figure 2. There is an

[22] If some components are not connected, the defined graph is not connected. Disconnected graphs are best described through their connected components. Hence, we assume that all components are connected. In Section 5, we will allow components that are not connected for directed graphs.

alternating path between leaf 7 and leaf 1. Hence, they are adjacent vertices in G. On any path between 3 and 7, there are at least two consecutive edges of H_u, hence, 7 is not adjacent to 3. $\square$

Remark 4.3 : A graph consisting of a single edge is defined by a graph labelled tree one component of which is an edge. Otherwise, all single edge components can be eliminated by node-joinings. The resulting tree has no node of degree 2.

However in a graph-labelled tree $\mathcal{T}$ that defines a graph with at least 2 vertices, it may be useful to insert a component consisting of a single edge (cf. Example 4.12 below): consider an edge uv of T, replace it by two edges uw and wv where w is new node, and define the new component corresponding to w as consisting of a single edge.

Lemma 4.4 : Let $\mathcal{T}$ be a graph labelled tree and $G = G(\mathcal{T})$.

(1) For each vertex x of $S(\mathcal{T})$, the set $A(x) \cap L_T$ is not empty.

(2) Let x, y be distinct vertices of some component H. If xy is an edge of H, there is an alternating path between any leaf in $A(x)$ and any leaf of $A(y)$. Conversely, if an alternating path links a leaf in $A(x)$ and a leaf of $A(y)$, then this path goes through H, and more precisely, through x and y, and $xy \in E_H$. Each component is isomorphic to an induced subgraph of G.

(3) Let uv be an edge of T. There is an alternating path between any leaf in $A(x)$ where x is a neighbour of $\rho_u(uv)$ in H_u and any leaf in $A(y)$ where y is a neighbour of $\rho_v(uv)$ in H_v. Any such path goes through the edge $\rho_u(uv)\rho_v(uv)$.

Proof: (1) Let x belong to H_u. Then $x = \rho_u(uv)$ for some (unique) edge uv of T. We use induction on the cardinality of $N_{T,u\setminus v}$, defined as the set of nodes of the connected component of $T - u$ that contains v.

If $\left|N_{T,u\setminus v}\right| = 1$, then v is a leaf and $A(x) = \{v\} = \rho_v(uv)$ is a satisfying leaf.

Otherwise, H_v has an edge $\rho_v(uv)y$ and then, $y = \rho_v(vw)$ for some edge vw of T. Then $N_{T,v\setminus w} \subset N_{T,u\setminus v}$ and $A(y) \cap L_T \subseteq A(x) \cap L_T$. The set $A(y) \cap L_T$ is not empty by induction, so is $A(x) \cap L_T$.

(2) Let $xy = \rho_u(uv)\rho_u(uw)$ be an edge of component H_u. Let $z \in A(x) \cap L_T$ and $z' \in A(y) \cap L_T$. By connecting alternating paths between z and x, and z' and y with the edge xy, we get an alternating path between z and z'. Any such path must through xy as one checks from the definitions.

For each vertex x of H_u, choose a vertex $\widehat{x}$ of G in $A(x) \cap L_T$. By the previous observations, the induced subgraph of G whose vertices are the $\widehat{x}$'s is isomorphic to H_u.

(3) The proof is similar to that of (2). $\square$

The following corollary illustrates these notions in the basic case of trees.

Corollary 4.5 : $G(\mathcal{T})$ is a tree if and only if :

(1) each component of $\mathcal{T}$ is a tree, and

(2) if $e = uv \in E_T$ is not a pendant edge, then at least one of $\rho_u(e)$ and/or $\rho_v(e)$ is a leaf of, respectively H_u and/or H_v.

Proof : Assume $G(\mathcal{T})$ is a tree. By Lemma 4.4(2), each component of $\mathcal{T}$ is isomorphic to an induced subgraph of $G(\mathcal{T})$, hence is a tree since components are connected. For Property (2), if none of $\rho_u(e)$ and $\rho_v(e)$ is a leaf, then the node-joining of u and v (Definition 4.1(d)) merges H_u and H_v into a component that contains a complete bipartite graph $K_{r,s}$ such that $r, s \geq 2$. This component has a cycle and $G(\mathcal{T})$ is not a tree by (1).

Conversely, let $\mathcal{T}$ satisfy (1) and (2). Consider $e = uv$ satisfying Property (2) : if we apply to it the node-joining operation, the component H'_w created in this way is still a tree. The obtained graph-labelled tree $\mathcal{T}'$ satisfies (1) and (2) and describes $G(\mathcal{T})$. By repeating this operation until all edges are pendant, we obtain a graph-labelled tree that defines $G(\mathcal{T})$ and that has one component that is a tree, all others being leaves. Hence $G(\mathcal{T})$ is a tree. □

A tree may be be defined from several nonisomorphic graph-labelled trees. A stronger condition than (2) of Corollary 4.5, namely Condition (2) of Theorem 4.7, yields unicity, up to isomorphism, of the graph-labelled trees that define trees.

Definition 4.6 : *The (canonical) split decomposition.*

(1) A *split* of a graph G is a bipartition of V_G into two sets V_1 and V_2 having each at least 2 vertices, such that the edges between V_1 and V_2 form a complete bipartite graph with at least one edge. Its bipartition is (A_1, A_2) where $A_1 \subseteq V_1$ and $A_2 \subseteq V_2$. Hence, G is $\kappa(G_1 \otimes G_2)$ for the induced subgraphs $G_1 = G[V_1]$ and $G_2 = G[V_1]$ with appropriate labelling in $D = \{\top, \bot\}$: the vertices in $A_1 \cup A_2$ are labelled by $\top$ and the others by $\bot$.

(2) A graph is defined as *prime*[23] if it has at least 4 vertices and no split. The connected graphs with 3 vertices are stars S_3 and triangles (i.e., K_3). A *star* S_n, $n \geq 3$, has a *center* and $n-1$ adjacent vertices that are leaves (S_n is a tree). Stars and cliques are not prime. A prime graph has actually at least 5 vertices. The cycles C_n for $n \geq 5$ are prime. Theorem 3 of [18], also proved as Theorem 2.9 in [25], states the following.

Theorem 4.7 : Every connected graph with at least 3 vertices is $G(\mathcal{T})$ for a unique graph-labelled tree $\mathcal{T}$ such that :

(1) each component H_v is singleton, or prime, or is a clique K_n or a star S_n, for some $n \geq 3$,

(2) if $e = uv \in E_T$, then H_u and H_v are not both cliques, and, if they are both stars, $\rho_u(e)$ and $\rho_v(e)$ are both centers or both leaves.

Unicity is understood up to isomorphism. □

[23] A different notion of prime graph is used in the theory of modular decomposition.

Such a graph-labelled tree is called *the split decomposition* of G. It can be obtained from an arbitrary graph-labelled tree that defines G by the node-splittings and the node-joinings of Definition 4.1(d) (see [25] for details). The resulting tree has no node of degree 2 (cf. Remark 4.3). We will actually describe graphs by graph-labelled trees that need not be canonical.

Examples 4.8 : (1) The split decomposition of a clique $K_n, n \geq 3$, has a unique component that is a clique. But any graph-labelled tree all components of which are cliques defines a clique. A clique $K_n, n \geq 3$ can be defined by a graph labelled tree all components of which are triangles or isolated vertices. By using node splitting, we can replace a component K_{r+s+2} where $r, s > 1$ by two components K_{r+1} and K_{s+1}.

(2) In the split decomposition of a tree, all components are stars, and if $e = uv \in E_T$, then $\rho_u(e)$ and $\rho_v(e)$ are both leaves of H_u and H_v by Corollary 4.5 and Theorem 4.7. Every tree with at least 3 nodes can be defined by a graph-labelled tree all components of which are stars S_3 or isolated vertices. As above, this can be achieved by node-splitting.

(3) A graph-labelled tree defines a DH graph if and only if all its components are stars and cliques ([25], Section 3.1).

(4) The article [25] also studies particular DH graphs called *3-leaf powers*. A 3-leaf power is a graph G defined as follows from a tree R : $V_G := L_R$ and two vertices are adjacent if and only if they are at distance at most 3 in R. A graph is a 3-leaf power if and only if it is obtained from a tree by substitutions of cliques to its vertices [4]. It follows that a graph G is a 3-leaf power if and only if it is a clique or is $G(\mathcal{T})$ for some graph-labelled tree $\mathcal{T}$ having one component, say H_0, that is a tree and all others are cliques (an isolated vertex is a clique K_1). Hence, the set L of 3-leaf powers is defined, up to labels, by the two equations:

$$L = K \cup \Lambda(L, L)$$
$$K = \top \cup (K \otimes K).$$

The set K is that of cliques where all vertices are live. The equation for L is derived from the first equation for rooted trees in Example 2.6(2). In the characterization of 3-leaf powers of [25], the component H_0 that is a tree is decomposed in the canonical way of Theorem 4.7. $\square$

Graph-labelled trees and substitution operations.

We will use D-graphs, where $D := \{\top, \bot\}$.

Definitions 4.9 : *Rooted graph-labelled trees and related notions.*

(a) Let $\mathcal{T}$ be a graph-labelled tree, with underlying tree T, not reduced to a single node. Let us select a node $r \in N_T$ that is not a leaf and make it a root for T. We call then $\mathcal{T}$ a *rooted graph-labelled* tree.

(b) If $u \in N_T$, we let $N_u := \{x \in N_T \mid x \leq_T u\}$ and $V_u := \{x \in V_G \mid x \leq_T u\}$. Hence, $V_r = L_T$, $V_u = \{u\}$ if $u \in L_T$.

(c) If $u \in N_T$ has father w, we say that $\rho_u(wu)$ is the *leader* of H_u, denoted by $\overline{u}$, and we define H'_u as the D-graph $H_u - \overline{u}$ where a vertex x live (labelled by $\top$) if it is adjacent to $\overline{u}$ in H_u and is dead (labelled by $\bot$) otherwise. Otherwise, u is the root r and $H'_r := H_r$, all its vertices being defined as dead.

(d) If $u \in N_T$, we define G_u as $G[V_u]$ labelled as follows:

(d.1) if $u = r$, then every vertex of $G_u = G$ is labelled as dead,

(d.2) if $u \neq r$, a vertex y of G_u is labelled by $\top$ if it is in $A(z)$ for some neighbour z (in H_u) of the leader of H_u; otherwise, it is labelled by $\bot$.

Example : Consider the graph-labelled tree of Figure 1 with root x. The vertices 4 and 5 are live in G_v, but not in G_u (so is 3). The vertices 2,6,7 are live in G_u. □

The following lemma relates graph-labelled trees and substitution operations. Note that the graphs H'_u depend on the root that is chosen.

Lemma 4.10 : Let $\mathcal{T}$ be a graph-labelled tree that defines G. If $u \in N_T - L_T$ has sons $u_1, ..., u_p$, and the corresponding p vertices of H'_u are $x_1, ..., x_p$ (that is, $x_i := \rho_u(\, uu_i)$), then we have $G_u = H'_u[G_{u_1}/x_1, ..., G_{u_p}/x_p]$.

Proof : Let $K := H'_u[G_{u_1}/x_1, ..., G_{u_p}/x_p]$.

1) The vertex set of G_u is $V_u := \{x \in V_G \mid x \leq_T u\}$ ($V_G = L_T$) hence, is the union the sets $V_{u_i} := \{x \in V_G \mid x \leq_T u_i\}$ that are the vertex sets of the graphs G_{u_i}. Hence, G_u and K have the same vertices.

2) As the graphs G_w are induced subgraphs of G, two vertices of G_{u_i} are adjacent in G_u if and only if they are in G as well as in G_{u_i}, hence also in K.

Consider vertices x of G_{u_i} and y of G_{u_j}, $j \neq i$. If they are adjacent in G_u, hence in G, they are linked by an alternating path, that must go through H_u (and not through its leader) and use its edge $\rho_u(u_iu)\rho_u(u_ju) = x_ix_j$, an edge of H'_u. This path goes through the leader $\rho_{u_i}(u_iu)$ of H_{u_i}. Hence, x is live in G_{u_i}. Similarly, y is live in G_{u_j}. Hence xy is an edge of K.

Conversely, if $xy \in E_K$, then x_ix_j is an edge of H'_u, the vertex x is live in G_{u_i} and y is live in G_{u_j}. Going back to definitions, we have an alternating path between x and y. Hence, xy is a edge of G hence of G_u.

3) If u is the root, all vertices of H'_u are dead, hence, so are those of K, as well as those of $G = G_r$.

Otherwise, let x be a vertex of G_{u_i} that is live. Hence, there an alterning path P between x and the leader $\rho_{u_i}(u_iu)$ of H_{u_i}. If $\rho_u(u_iu) = x_i$ is a neighbour of the leader z of H_u, then x is live in K. It is also in G_u because P can be extended into an alterning path from x to z. If $\rho_u(u_iu) = x_i$ is not a neighbour of z, then x is dead in K. It is also in G_u because P cannot be extended into an alterning path from x to z.

If x is not live in G_{u_i}, then it is not in K, and is not either in G_u, because otherwise, the alternating path between x and z would give a path P as above. □

If $\mathcal{M}$ is a finite set of connected (unlabelled) graphs having at least 2 vertices. We define $\mathcal{G}(\mathcal{M})$ as the set of graphs described by graph-labelled trees whose components are in $\mathcal{M}$. For each $H \in \mathcal{M}$ and $x \in V_H$, we define $H\backslash x$ as the D-graph $H - x$ where the neighbours of x are labelled by $\top$ and the others by $\bot$, hence are defined as dead. Note that $H - x$ need not be connected. The labelled graph H'_u of Definition 4.9 is $H_u\backslash\overline{u}$.

In the following theorem, σ denotes operations $\sigma[H, x_1, ..., x_p]$ for $H \in \mathcal{M}$, where all vertices of H are dead, and σ' operations $\sigma[H\backslash x, x_1, ..., x_p]$ for $H \in \mathcal{M}$ and $x \in V_H$.

Theorem 4.11 : If $\mathcal{M}$ is a finite set of connected graphs having at least 2 vertices, then $\mathcal{G}(\mathcal{M})$ is the set S defined by the two equations :

$$S = \kappa(\top) \cup ... \cup \ \sigma(U, ..., U) \cup ...$$
$$U = \top \cup ... \cup \ \sigma'(U, ..., U) \cup ...$$

Proof : Let G belong to S. If it is a dead isolated vertex $\kappa(\top)$, it is in $\mathcal{G}(\mathcal{M})$. Otherwise, it is defined by a finite term $t = \sigma(t_1, ..., t_p)$ where $t_1, ..., t_p$ are terms over $\top$ and the operations σ'. By Lemma 4.10, this term represents a rooted graph labelled tree T. The component at the root is isomorphic to H such that $\sigma = \sigma[H, x_1, ..., x_p]$. The terms $t_1, ..., t_p$ represent the subtrees of T issued from the p sons of the root. Hence, G is described by a rooted graph-labelled tree with components in $\mathcal{M}$. Hence $G \in \mathcal{G}(\mathcal{M})$.

Let conversely $G \in \mathcal{G}(\mathcal{M})$. If it is a dead isolated vertex, then it is in S, defined by $\kappa(\top)$. Otherwise it is defined by a rooted graph-labelled tree T, hence, by a term $t = \sigma(t_1, ..., t_p)$ as above. The subtrees of T issued from the sons of the root are defined by the terms $t_1, ..., t_p$. $\square$

Examples 4.12 : (1) We examine some of the operations $\sigma[H, x_1, ..., x_p]$ and $\sigma[H\backslash x, x_1, ..., x_{p-1}]$ that arise in split decompositions, in particular, in the canonical ones.

Case 1: H or $H\backslash x$ is a clique $K_p, p \geq 1$, all vertices being live. Then, we have :

$$\sigma[K_p, x_1, ..., x_p](G_1, ..., G_p) = G_1 \otimes ... \otimes G_p.$$

Case 2 : $H = S_p, p \geq 3$, with center x_1, that is live in $S_p\backslash x_p$, all other vertices being dead. We have :

$$\sigma[S_p\backslash x_p, x_1, ..., x_{p-1}](G_1, ..., G_{p-1}) = \Lambda(\Lambda(...\Lambda(G_1, G_2), G_3), ..., G_{p-1}))...)).$$

Case 3 : $H = S_p, p \geq 3$, with center x_p. Then, all vertices of $S_p\backslash x_p$ are live and we have :

$$\sigma[S_p\backslash x_p, x_1, ..., x_{p-1}](G_1, ..., G_{p-1}) = G_1 \oplus ... \oplus G_{p-1}.$$

(2) A connected DH graph is defined by a graph-labelled tree $\mathcal{T}$ whose components are stars, cliques and single vertices. By means of node splittings (Definition 4.1(d)), we can transform $\mathcal{T}$ into a graph-labelled tree whose components are stars S_3, triangles K_3, together with one component[24] K_2 : for $n \geq 4$, a component (isomorphic to) K_n can be split into K_3 and K_{n-1}, and a component S_n can be split into S_3 and S_{n-1} where the center of S_3 is linked to a leaf of S_{n-1}.

Let us take as root the component K_2. We obtain from Theorem 4.11 the following equations for defining these graphs :

$S = \kappa(\top) \cup \ \kappa(U \otimes U)$,

$U = \top \cup (U \oplus U) \cup (U \otimes U) \cup \Lambda(U, U)$,

that are the two equations of Proposition 2.7(2). □

Clique-width bounds from graph-labelled trees

Theorem 4.13 : Let G be a connected graph defined by rooted graph labelled tree $\mathcal{T}$ such that each operation $\sigma[H'_u, x_1, ..., x_p]$ has width at most k. Then $cwd^{\perp}(G) \leq k$ and $cwd(G) \leq k + 1$.

Proof: Immediate consequence of Proposition 3.2 and Theorem 4.11. □

The next lemma gives an upper-bound to the widths of the terms in $T_{\perp}(F_C)$ that define the D-graphs H'_u.

Lemma 4.14 : Let H be a D-graph of clique-width k. Then $cwd^{\perp}(H) \leq k + \min\{k, \left|V_H^{live}\right|, \left|V_H^{dead}\right|\}$.

Proof: That $cwd^{\perp}(H) \leq 2k$ follows from Lemma 1.3. This lemma uses a partition of the vertex set in two parts. At the end of the construction, all vertices of one part are relabelled into $\perp$.

For proving that $cwd^{\perp}(H) \leq k + \left|V_H^{live}\right|$, we consider a term that defines H with a set C' of k labels different $\perp$. Assume that $V_H^{live} = \{x_1, ..., x_p\}$. We add labels $c_1, ..., c_p$ to C' such that c_i will only label x_i.

We transform t into t' accordingly. In particular, to take a typical case, if at some position in t the operation $add_{a,b}$ adds edges between x_i, labelled at this point by a (there may be other a-labelled vertices) and b-labelled vertices, then we replace it by $add_{c_i,b} \circ add_{a,b}$. Hence, we can use $p + k$ labels different from $\perp$.

If $V_H^{dead} = \{x_1, ..., x_p\}$. We do a similar construction. □

Theorem 4.15 : Let G be defined by a canonical graph-labelled tree $\mathcal{T}$ whose components have maximal clique-width m and maximal degree d, then $m \leq cwd(G) \leq m + \min\{m, d\} + 1$.

Proof : The inequality $m \leq cwd(G)$ follows from Lemma 4.4(2) since clique-width is monotone with respect to the induced subgraph relation.

We now prove the other inequality. For each component H_u, the number of live vertices in H'_u is at most d. Hence, Lemma 4.14 gives $cwd^{\perp}(H'_u) \leq$

[24] By Remark 4.3.

$m + \min\{m, d\}$. Then, Theorem 4.13 gives $cwd^{\perp}(G) \leq m + \min\{m, d\}$ hence, $cwd(G) \leq m + \min\{m, d\}+1$. $\square$

Remark 4.16 : If G is a tree that is not a star and is defined by $\mathcal{T}$ whose components are stars with three nodes, hence of clique-width 2, we have $m = 2$ and $cwd(G) = 3$.

Parity graphs.

A graph is a *parity graph* if for any two vertices, the induced paths joining them have the same parity. DH graphs and bipartite graphs are parity graphs. The article [6] establishes that parity graphs are those with a split decompositions whose components are cliques and bipartite graphs. We do not get a finite grammar as bipartite graphs, whence also parity graphs, have unbounded clique-width.

Rank-width.

Rank-width is another complexity measure initially defined for undirected graphs[25], denoted by $rwd(G)$. As tree-width and clique-width, it is based on a tree-structuring of the given graph G. The DH graphs are those of rank-width 1 [35]. Rank-width is related to clique-width by the inequalities $rwd(G) \leq cwd(G) \leq 2^{rwd(G)+1} - 1$. Furthermore, if $G = G(\mathcal{T})$ for some graph-labelled tree $\mathcal{T}$, and m is the maximal rank-width of a component H_u, then $rwd(G) = m$ (Theorem 4.3 of [31]). Hence, if $cwd(H_u) \leq m$ for all u, we get $rwd(G) \leq m$ and $cwd(G) \leq 2^{m+1} - 1$. The bound given by Theorem 4.15 is thus better.

5 Directed graphs

We now extend our results to directed graphs. We recall that Cunnigham defines in [18] a canonical split decomposition for directed graphs that are strongly connected (Theorem 1). An undirected graph can be seen as a directed one where each edge has an opposite one. It is connected if and only if the corresponding directed graph is strongly connected. Hence, Theorem 4.7 is a special case of a more general one for directed graphs[26].

We use again graph-labelled trees and split decomposition graphs as in [8]. In order to extend to directed graphs the results of Section 4, we will revise the notion of substitution of Section 2 for graphs with labels that encode edge directions. In some sense, the set of live vertices is partitioned into three sets, designated by the tags $\top, +, -$, attached to the labels of the sets C used to construct graphs.

[25] Extension to directed graphs is in [32].

[26] We thought better to begin with undirected graphs, because the formal setting is simpler and most graph structure theory and graph algorithms concern undirected graphs.

In this section, all graphs are directed unless otherwise indicated. However, the graphs representing graph-labelled trees have undirected edges as well as directed ones.

Definition 5.1 : *Substitutions.*
We let D be the set of labels $\{\bot, \top, +, -\}$ ordered in such a way that $\bot < + < \top$, $\bot < - < \top$ and $+$ and $-$ are incomparable. Let K be a D-graph with vertex set $\{x_1, \ldots, x_p\}$ and $H_1, \ldots, H_p$ be pairwise disjoint D-graphs, that are disjoint from K. We define a D-graph G as follows:

$G := K[H_1/x_1, \ldots, H_p/x_p]$ where
$V_G := V_{H_1} \uplus \ldots \uplus V_{H_p}$,
$\pi_G(v) := \inf\{\pi_{H_i}(v), \pi_K(x_i)\}$ if $v \in V_{H_i}$,

its (directed) edges are as follows, for $u, v \in V_G$:

$uv \in E_G$ if and only if :
$uv \in E_{H_i}$ for some i,
or $\pi_{H_i}(u) \in \{\top, +\}$, $\pi_{H_j}(v) \in \{\top, -\}$ and $x_i x_j \in E_K$ (and so $i \neq j$).

Lemma 5.12 motivates this definition. If we consider an undirected graph as a directed graph where each edge has an opposite one, and vertices are labelled by $\bot$ or $\top$, then Definition 5.1 gives the same notion of substitution as Definition 2.1.

Example : Let K be $x \longrightarrow y \longleftarrow z$. where the labels of x, y and z are respectively $+, -$ and $\top$. Let X be the edgeless D-graph $\bot(0) \oplus +(1) \oplus -(2) \oplus \top(3)$ (its vertices 0,1,2,3 are labelled respectively by $\bot, +, -, \top$). Let similarly $Y := +(4) \oplus -(5) \oplus \top(6)$ and $Z := +(7) \oplus -(8) \oplus \top(9)$.The graph $K[X/x, Y/y, Z/z]$ is shown in Figure 3. The labels of vertices 0,1,5,7,8 and 9 are as in X, Y and Z. Those of 2,3,4 and 6 are respectively $\bot$, $+$, $\bot$ and $-$ because of the inf operator in Definition 5.1(a). $\square$

We obtain graph operations, as in Section 2, that we will use to describe the directed graphs defined by graph-labelled trees.

Definition 5.2: *Graph operations based on substitution.*
For each D-graph K with vertex set enumerated as $\{x_1, \ldots, x_p\}$, we define as follows a p-ary operation on D-graphs denoted by $\sigma[K, x_1, \ldots, x_p]$:

$$\sigma[K, x_1, \ldots, x_p](H_1, \ldots, H_p) := K[H_1/x_1, \ldots, H_p/x_p]$$

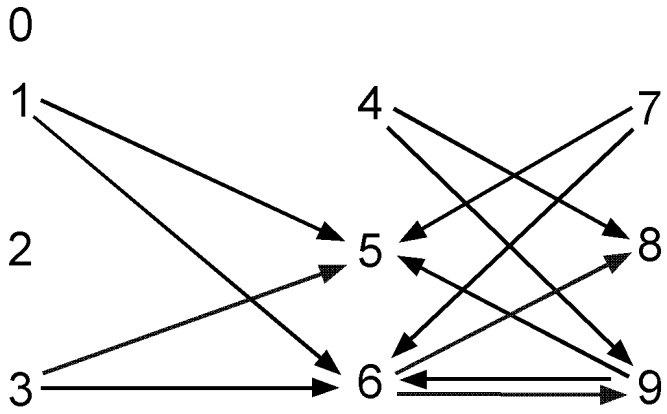

Figure 3: Figure 3

where $H_1, \ldots, H_p$ are pairwise disjoint and disjoint from K. If they are not, we replace them by isomorphic copies, so that $\sigma[K, x_1, \ldots, x_p]$ becomes a p-ary operation on abstract D-graphs. $\square$

If we extend Definition 5.1 so that K has other vertices than $x_1, \ldots, x_p$, then the properties of Proposition 2.2 are still valid.

Substitutions as derived operations

Clique-width is defined for directed graphs with the help of the operations $\overrightarrow{add}_{a,b}$ that create directed edges from a-labelled vertices to b-labelled ones. Our objective is to bound the clique-width of $G := K[H_1/x_1, \ldots, H_p/x_p]$ as $O(\max\{cwd(K), cwd(H_1), \ldots, cwd(H_p)\})$.

Definitions and notations 5.3. We consider $G := K[H_1/x_1, \ldots, H_p/x_p]$.

(a) The graphs $G, K, H_1, \ldots, H_p$ are D-graphs that we will define by terms t_G, t_K, ... in $T(F_E)$ where $E \subset C \times D \times D$, in such a way that $G = relab_f(val(t_G))$, where $f((a, \alpha, \beta)) := \beta$ for all (a, α, β) in E, and similarly for $K, H_1, \ldots, H_p$, with the same function f. More precisely, we will use $E := \{(a, \alpha, \beta) \mid a \in C, \alpha, \beta \in D, \beta \leq \alpha\}$. (The set D is ordered, cf. Definition 5.1).

(b) Let $K \simeq val(t)$ for some term $t_K \in T(F_C)$. (The labels in C do not specify the labelling $\pi_K : V_K \to D$). Let $x_1, \ldots, x_p$ be the vertices of K. Each vertex x_i is defined by a nullary symbol $\boldsymbol{a_i}(x_i)$ in t. We construct a term $\widehat{t}_K$ in $T(F_E, \{u_1, \ldots, u_p\})$ as follows :

- we replace each $\boldsymbol{a_i}(x_i)$ by the variable u_i,

- we replace each operation $relab_h$ by $relab_{\widehat{h}}$ where $\widehat{h}((a, \alpha, \beta)) := (h(a), \alpha, \beta)$ for all $(a, \alpha, \beta) \in E$,

- we replace each operation $\overrightarrow{add}_{a,b}$ by the composition (in any order) of the operations $\overrightarrow{add}_{(a,\alpha,\beta),(b,\alpha',\beta')}$ such that $\alpha \in \{\top, +\}, \alpha' \in \{\top, -\}$ and $\beta, \beta' \in D$.

Furthermore, for each $i := 1, ..., p$, we define $h_i : E \to E$ by $h_i((a, \alpha, \beta)) := ((a_i, \beta, \inf\{\beta, \pi_K(x_i)\}))$ for $(a, \alpha, \beta) \in E$.$\square$

With these definitions and notation, we have :

Lemma 5.4 : If for each i we have a term $t_i \in T(F_E)$ such that $H_i = relab_f(val(t_i))$, then :

$$K[H_1/x_1, \ldots, H_p/x_p] = relab_f(val(\widehat{t_K}\ [relab_{h_1}(t_1)/u_1, ..., relab_{h_p}(t_p)/u_p])).$$

Proof : Let $G := K[H_1/x_1, \ldots, H_p/x_p]$ and
$G' := relab_f(val(\widehat{t_K}[relab_{h_1}(t_1)/u_1, ..., relab_{h_p}(t_p)/u_p]))$.
These two graphs have the same vertex set, $V_{H_1} \uplus \ldots \uplus V_{H_p}$.

Let $u \in V_{H_i}$. Let (a, α, β) be its label in $val(t_i)$. Its label is β in H_i, and is $(a_i, \beta, \inf\{\beta, \pi_K(x_i)\})$ in $relab_{h_i}(t_i)$. Its label in G' is thus $\inf\{\beta, \pi_K(x_i)\}$ because the relabellings in $\widehat{t_K}$ do not modify the third components of labels. It is same in G by Definition 5.1.

We now compare the edges of G and of G'.

Case 1 : $u, v \in V_{H_i}$. If $uv \in E_{H_i}$, it is also an edge of G and of G'. If $uv \notin E_{H_i}$, it is not an edge of G either. And it is not an edge of G' because the labels of u and v have the same first components in $relab_{h_i}(t_i)$ and the relabellings in $\widehat{t_K}$ maintain this equality. Hence, no edge between u and v is created by the operations $\overrightarrow{add}_{(a,\alpha,\beta),(b,\alpha',\beta')}$ of $\widehat{t_K}$ (in which $a \neq b$). Hence, $uv \notin E_{G'}$.

Case 2 : $u \in V_{H_i}, v \in V_{H_j}$, $i \neq j$. If $uv \in E_G$, then $x_i x_j$ is an edge of K and the labels of u and v in H_i and H_j are respectively $\alpha \in \{\top, +\}$ and $\alpha' \in \{\top, -\}$ by Definition 5.1.

Let us now consider G'. At some position w in t_K the edge $x_i x_j$ is created by an operation $\overrightarrow{add}_{a,b}$.

In $val(relab_{h_i}(t_i))$ and $val(relab_{h_j}(t_j))$, the labels of u and v are respectively (a_i, α, β) and (a_j, α', β'). for some β and β', and, in t_K, a_i and a_j are relabelled into a and b at position w. The labels of u and v are thus (a, α, β) and (b, α', β') in $\widehat{t_K}$ at the position p' corresponding to p. Hence, uv is an edge of G', created by $\overrightarrow{add}_{(a,\alpha,\beta),(b,\alpha',\beta')}$ at p'.

Hence every edge of G is an edge of G'. The proof is similar in the other direction. Hence, $G = G'$. $\square$

No label $(a, \bot, \bot)$ occurs in an edge addition operation of $\widehat{t_K}$. Furthermore, $f((a, \bot, \bot)) = \bot$ for each $a \in C$. Hence, all labels $(a, \bot, \bot)$ can be replaced by the unique label $(c, \bot, \bot)$ for some fixed $c \in C$. It follows that $\widehat{t_K}$ and the relabellings h_i only use the following $8\,|C| + 1$ labels :

$(c, \bot, \bot)$,

$(a, \top, \top), (a, \top, +), (a, \top, -), (a, \top, \bot),$
$(a, +, +), (a, +, \bot),$
$(a, -, -), (a, -, \bot),$

for all $a \in C$. By using this remark, we obtain:

Proposition 5.5 : If a graph G is defined up to vertex labels by a composition of operations $\sigma[K, x_1, \ldots, x_p]$ such that each graph K has clique-width at most k, then, $cwd(G) \leq 8k + 1$.

Proof: Let G be defined by a term over operations $\sigma[K, x_1, \ldots, x_p]$ such that $cwd(K) \leq k$, and, of course, nullary symbols. The terms t_K can be written with the labels of a set C of k labels; as nullary symbols one can use the single one $\boldsymbol{c}$.

By composing the terms $\widehat{t_K}$ and the relabellings h_i according to Lemma 5.4, we obtain a term that defines G by using only $8k + 1$ labels. Hence, $cwd(G) \leq 8k + 1$. $\square$

Directed graph-labelled trees.

Definition 5.6 : *Graph-labelled trees describing directed graphs.*

In the following definition, unspecified definitions and notation are as in Definition 4.1. We avoid repetions as much as possible.

(a) A *directed graph-labelled tree*, denoted by $\mathcal{T}$ is an undirected tree T with at least 3 nodes, that is equipped, for each node $v \in N_T$, with a directed graph H_v, called a *component*, and a bijection $\rho_v : Inc_T(v) \to V_{H_v}$. Components are pairwise disjoint and need not be connected, see Remark 5.7(2) below. If v is a leaf, then H_v has a single vertex that we identify with v.

(b) We define $S(\mathcal{T})$ as the graph consisting of the union of the components together with the undirected edges $\rho_u(e)\rho_v(e)$ for $e = uv$, (cf. Figure 4). The directed edges are those inside the components. A path, or a walk[27], in $S(\mathcal{T})$ from x to y is *alternating* if no two consecutive edges are in a same component and all edges belonging to components are directed from x to y. There is at most one alternating path from x to y, but there may exist also one from y to x. This is the case if and only if each directed edge zz' on this path has an opposite edge $z'z$.

To be more precise, we define for a vertex x of a component H_u, $A^-(x)$ as the set of vertices of $S(\mathcal{T})$ accessible from x by an alternating path whose first edge is not in H_u, and $A^+(x)$ as the set of vertices w of $S(\mathcal{T})$ such that there is an alternating path from w to x whose last edge is not in H_u.

(c) The *graph described by* $\mathcal{T}$, denoted by $G(\mathcal{T})$, has vertex set L_T (the set of leaves of T) and an edge uv if and only if there is an alternating path from

[27] A *walk* is like a path but it can go several times through a vertex.

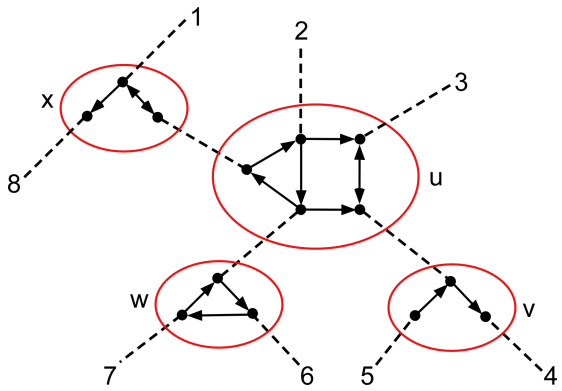

Figure 4: A directed graph-labelled tree $\mathcal{T}$.

u to v. If there is a path $u_1 \to u_2 ... \to u_p$ in $G(\mathcal{T})$, the concatenation of the alternating paths corresponding to the edges $u_i u_{i+1}$ forms an alternating walk from u_1 to u_p.

(d) The *node-joining operation* (called *elimination* of an edge e of T in [8]) is defined as follows. If $e = uv$ is an edge between two internal nodes of T, its contraction fuses u and v into a single node say w, giving tree T', and replaces the graph $H_u \uplus H_v$ by $H'_w := (H_u \uplus H_v) - \{\rho_u(e), \rho_v(e)\}$ augmented with an edge from any vertex x in H_u to any vertex y in H_v such that $x\rho_u(e) \in E_{H_u}$ and $\rho_v(e)y \in E_{H_v}$. We obtain a directed graph-labelled tree $\mathcal{T}'$ that describes the same graph : this is easy to check by considering alternating paths. By iterating as much as possible this elimination step, we obtain a star whose central component is isomorphic to $G(\mathcal{T})$.

The opposite transformation called *node-splitting* also preserves the defined graph. □

The notion of canonical decomposition for strongly connected (directed) graphs (Theorem 2 of [18]) uses cliques, stars and particular graphs called *cycles of transitive tournaments* that have clique-width at most 4 (by Proposition 4.16 of [8]). We will not use this difficult notion. We will describe directed graphs by directed graph-labelled trees, either canonical or not. Our results will be valid even if some components are not connected.

Examples 5.7 : (1) Figures 4 shows a directed graph-labelled tree $\mathcal{T}$ and Figure 5 the corresponding graph $G(\mathcal{T})$. The double arrow in the components H_x and H_u indicate pairs of opposite directed edges. For a comparison with Figure 1, there is no alternating path between 2 and 7, in either direction. Hence, these two vertices are not adjacent in $G(\mathcal{T})$.

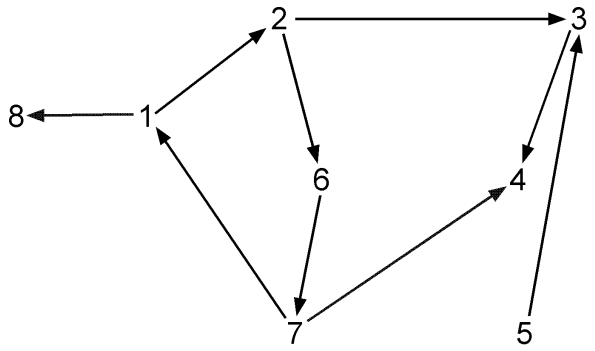

Figure 5: The graph defined by $\mathcal{T}$ of Figure 4

(2) A directed graph-labelled tree may define a disconnected graph although its components are connected. As a small example, consider $S(\mathcal{T})$ defined as $x - z' \longrightarrow z - u \longleftarrow u' - y$: its internal components are $z' \longrightarrow z$ and $u \longleftarrow u'$. Then $G(\mathcal{T})$ consists of the two isolated vertices x and y. Eliminating the edge between the two internal components yields a component consisting of two isolated vertices u' and z'.

Proposition 5.8 : Let G be defined by a directed graph-labelled tree $\mathcal{T}$ whose components are strongly connected.

(1) For each vertex x of $S(\mathcal{T})$, we have $A^+(x) \cap L_T \neq \emptyset$ and $A^-(x) \cap L_T \neq \emptyset$.

(2) If xy is an edge of H_u, then $zz' \in E_G$ for all $z \in A^+(x) \cap L_T$ and $z' \in A^-(y) \cap L_T$. Conversely, if $x = \rho_u(uv)$ and $y = \rho_u(uw)$ are distinct vertices of H_u, if $z \in L_T \cap N_{T,u\setminus v}$, $z' \in L_T \cap N_{T,u\setminus w}$ and $zz' \in E_G$, then $z \in A^+(x) \cap L_T$, $z' \in A^-(y) \cap L_T$ and xy is an edge of H_u.

(3) G is strongly connected.

Note that T has at least two leaves and so, that G has at least two vertices.

Proof : (1) Let x be a vertex of a component H_u. Hence, $x = \rho_u(uv)$ for some node v. We use induction on the cardinality of $N_{T,u\setminus v}$ (cf. the proof of Lemma 4.4(1)). If it is 1, then v is a leaf and $A^+(x) \cap L_T = A^-(x) \cap L_T = \{v\}$.

Otherwise, we have in H_v an edge zy such that $y = \rho_v(uv)$ and $z = \rho_v(vw)$ for some edge vw of T. We have $N_{T,v\setminus w} \subset N_{T,u\setminus v}$, hence, by induction, $A^+(z) \cap L_T \neq \emptyset$ and $A^+(z) \cap L_T \subseteq A^+(x) \cap L_T$ which proves $A^+(x) \cap L_T \neq \emptyset$. The proof that $A^-(x) \cap L_T \neq \emptyset$ is similar.

(2) Just consider alternating paths, as in the proof of Lemma 4.4.

(3) The node-joining operation preserves the strong connectedness of the components as one checks from Definition 5.6(d). By repeating this operation, one obtains a directed graph-labelled tree that defines G and consists of one

"central" strongly connected component and leaves. This component is isomorphic to G, hence G is strongly connected. $\square$

Proposition 5.9 : Let G be defined by a graph-labelled tree $\mathcal{T}$. Then G is strongly connected if and only if all components of $\mathcal{T}$ are strongly connected.

Proof : The "if" direction is proved in the previous proposition. For the converse, let x and y be distinct vertices of a component H_u. Let $s \in A^+(x) \cap L_T$ and $t \in A^-(y) \cap L_T$. There is a path in G from s to t. Each edge of this path corresponds to an alternating path in $S(\mathcal{T})$ whose directed edges are oriented from s to t. The concatenation of these paths is a walk[28] in $S(\mathcal{T})$. It is not necessarily a path because some undirected edges may be traversed twice. (In the example of Figure 4 the path $2 \longrightarrow 6 \longrightarrow 7 \longrightarrow 4$ comes from a walk that traverses twice the edge between H_u and H_w). This walk must enter H_u first via x and exit it last via y. Its edges belonging to H_u form a directed path from x to y. Hence, H_u is strongly connected. $\square$

Remark 5.10 : Even if $G(\mathcal{T})$ is strongly connected, some components of $\mathcal{T}$ may not be isomorphic to induced subgraphs of $G(\mathcal{T})$ (by contrast with Lemma 4.4(2)). Consider for an example a directed graph-labelled tree having two internal components isomorphic to the directed cycles $\overrightarrow{C}_3$. It defines $\overrightarrow{C}_4$ that does not contain $\overrightarrow{C}_3$ as an induced subgaph. We will compare below the clique-widths of a graph and its components.

Directed graph-labelled trees and substitution operations.

We will use the set of labels $D := \{\bot, +, -, \top\}$.

Definitions 5.11 : *Rooted graph-labelled trees and related notions.*
(a) Let $\mathcal{T}$ be a directed graph-labelled tree, with underlying tree T and $G := G(\mathcal{T})$. Let us select a node $r \in N_T - L_T$ and make it a root for T. If $u \in N_T$, then V_u is as in Definition 4.9.

(b) The *leader* of a component H_u such that u is not the root is the vertex $\rho_u(wu)$ denoted by $\overline{u}$, where w is the father of u. If $u \in N_T - L_T$, we define a D-graph H'_u as the follows :

if $u = r$, then $H'_r := H_r$, and all its vertices are dead ;

otherwise, we define $H'_u := H_u - \overline{u}$ where a vertex x is labelled as follows : its label is $\top$ if $x\overline{u}, \overline{u}x \in E_{H_u}$, it is $+$ if $x\overline{u} \in E_{H_u}$ and $\overline{u}x \notin E_{H_u}$, it is $-$ if $\overline{u}x \in E_{H_u}$ and $x\overline{u} \notin E_{H_u}$, and it is $\bot$ if $x\overline{u}$ and $\overline{u}x$ are not in E_{H_u}.

[28] A vertex may occur several times on a walk.

Note that the graphs H'_u depend on the chosen root r.

(c) If $u \in N_T$, we define $G_u := G[V_u]$ labelled as follows:
If $u = r$, all vertices of $G_u = G$ are dead.
Otherwise, a vertex x has label $\top$ if there are alternating paths from x to $\overline{u}$ and from $\overline{u}$ to x; it has label $+$ if there is an alternating path from x to $\overline{u}$ and no such path from $\overline{u}$ to x; it has label $-$ if there is an alternating path from $\overline{u}$ to x and no such path from x to $\overline{u}$ and label $\perp$ if there are no alternating paths between x and $\overline{u}$. $\square$

The following lemma extends Lemma 4.10.

Lemma 5.12 : If $u \in N_T - L_T$ has sons $u_1, ..., u_p$ and the corresponding p vertices of H'_u are $x_1, ..., x_p$ (that is, $x_i := \rho_u(\ uu_i)$), then we have $G_u = H'_u[G_{u_1}/x_1, ..., G_{u_p}/x_p]$.

Proof : Let $K := H'_u[G_{u_1}/x_1, ..., G_{u_p}/x_p]$. As in Lemma 4.10, the vertex sets of G_u and K are the same and the edges of G_{u_i} are the same as in G_u and K.

We consider x in G_{u_i} and y in G_{u_j}, $j \neq i$. If xy is an edge of G, there is an alternating path from x to y. It must go through H_u (and not through its leader) via the (directed) edge $\rho_u(u_iu)\rho_u(u_ju) = x_ix_j$ in H'_u. This path goes through the leader $\rho_{u_i}(u_iu)$ of H_{u_i}. Hence, x is live in G_{u_i}. More precisely, it has label $+$ or $\top$. Similarly, y has label $-$ or $\top$ in G_{u_j}. Hence xy is an edge of K.

Conversely, if $xy \in E_K$, then x_ix_j is an edge of H'_u, x has label $+$ or $\top$ in G_{u_i} and y has label $-$ or $\top$ in G_{u_j}. Going back to definitions, we have an alternating path from x and y built from alternating paths from x to x_i, and x_j to y and the edge x_ix_j. Hence, xy is a edge of G, hence of G_u.

It remains to compare the vertex labels in K and in G_u.

Let x be a vertex of G_{u_i} labelled by $+$ in G_u. There is an alternating path from x to the leader $\overline{u}$ of H_u and no such path from $\overline{u}$ to x. Hence, there is an alternating path from x to $\overline{u_i}$ and an edge in H_u from $\rho_u(uu_i)$ to $\overline{u}$. Hence the label of $\rho_u(uu_i)$ is either $+$ or $\top$. The label of x in G_{u_i} is either either $+$ or $\top$, hence its label in K is either $+$ or $\top$. If it would be $\top$, we would have an edge in H_u from $\overline{u}$ to $\rho_u(uu_i)$ and an alternating path from $\overline{u}$ to x and x would have label $\top$ in G_u. Hence x has label $+$ in K.

The proofs are similar for the other labels. $\square$

Theorem 5.13 : Let G be defined by directed graph-labelled tree $\mathcal{T}$ whose components have clique-width at most k. Then $cwd(G) \leq 8k + 1$.

Proof: From Proposition 5.5 and Lemma 5.12, along the lines of Theorem 4.11. $\square$

One can define a graph-labelled tree $\mathcal{T}$ that has components of arbitrary large clique-width but defines a graph without edges, hence of clique-width 1.

We build $\mathcal{T}$ with any connected graph H as "root" component. Attach to each vertex x of H a path of the form $x - v_x \longrightarrow v'_x - w'_x \longleftarrow w_x - \widetilde{x}$. Then $G(\mathcal{T})$ consists of the isolated vertices $\widetilde{x}$. For strongly connected graphs, we have a better situation.

Proposition 5.14 : There is a function f such that $cwd(H) \leq f(cwd(G))$ whenever G is strongly connected and H is a component of some directed graph-labelled tree that defines it.

We need some definitions. An edge xy of a graph G is *special* if x has out-degree 1 and y has indegree 1. Let F be a set special edges. A path with all its edges in F is called an *F-path*. The graph $G\backslash F$, obtained from G by contracting the edges of F is defined as follows, where X is the set of terminal ends of the edges of F :

$V_{G\backslash F} := V_G - X$,

$xy \in E_{G\backslash F}$ if and only if $x, y \notin X$ and, either $xy \in E_G$ or there exist a vertex z such that $zy \in E_G$ and an F-path from x to z.

If F forms a directed cycle (necessarily disconnected from $G - X$), then $V_{G\backslash F} = G - X$.

Lemma 5.15 : There is a function f such that $cwd(G\backslash F) \leq f(cwd(G))$ for every directed graph G and every set F of special edges.

Proof: There exists a monadic second-order transduction (not using edge set quantifications) that maps the pair (G, X) of a graph G and a set X that is the set of terminal ends of the edges of a set F of special edges (uniquely determined from X) to $G\backslash F$. Its definition is a straightforward translation from the definition.

The existence of f follows from of Corollary 7.38(2) of [14]. □

However, the proof does not give a good bound[29] for f. It is an open question whether $cwd(G\backslash F) \leq cwd(G)$ or even[30] $cwd(G\backslash F) = O(cwd(G))$.

Proof of Proposition 5.14: Let G be strongly connected defined by some graph-labelled tree $\mathcal{T}$. Let H_u be any component not reduced to a single vertex. We claim that there exists an induced subgraph G' of G such that H_u is isomorphic to $G'\backslash F$ for some set F of special edges.

Let $x := \rho_u(uv)$.

[29] For a comparison, if an undirected graph H is obtained from G by erasing degree 2 vertices, that is by contracting a set of edges that have all an end vertex of degree 2, then $cwd(H) \leq 2^{cwd(G)+1} - 1$ (Proposition 2 of [10]). However, we prove in [17] that $cwd(H) > 2^{cwd(G)/4}$ in some cases.

[30] We can prove that clique-width is not increased when a path of special edges from x to y is contracted into an edge xy. Otherwise, we can bound $cwd(G\backslash F)$ by using $f(k) =: k^2.3^{k-1}$ [17].

Case 1 : There exists a vertex z of G belonging to $A^+(x) \cap A^-(x) \cap L_T$. We choose one that we denote by $\widehat{x} := z$.

Case 2 : Otherwise, H_v has at least 3 vertices. Let $x' := \rho_u(uv)$. Then H_v contains a directed cycle going through x' that does not consist of two opposite edges incident to x', otherwise we are in Case 1. Hence there are edges $x'y$, $y'x'$ and a directed path from y to y' that avoids x'. Let $z \in A^-(y) \cap L_T$ and $z' \in A^+(y') \cap L_T$. There is a path in G from z to z'. Each of its edges corresponds to an alternating path in $S(\mathcal{T})$. By concatenating these paths, one gets an alternating walk, all vertices of which are in $N_{T,u\setminus v}$. We let P_x be such a path in G and we denote z by $\widehat{x}_1$ and z' by $\widehat{x}_2$. The paths P_x are pairwise disjoint.

Let X be the union of the vertex sets of the paths P_x for $x \in V_{H_u}$ together with the vertices $\widehat{x}$. Let $G' := G[X]$ and F be the set of edges of the paths P_x. They are special edges of G' because the paths are pairwise disjoint.

Let $H := G'\setminus F$. The two end vertices $\widehat{x}_1$ and $\widehat{x}_2$ of a path P_x get fused into a single vertex that we denote by $\widehat{x}$. The vertex set of H is thus $\{\widehat{x} \mid x \in V_{H_u}\}$.

Claim: The bijection $x \longmapsto \widehat{x}$ is an isomorphism $H \to H_u$.

Proof of the claim:

Let xy be an edge of H_u. Assume x is in Case 1 and y in Case 2. We have alternating paths from $\widehat{x}$ to x and from y to $\widehat{y}_1$, and one from $\widehat{x}$ to $\widehat{y}_1$. Hence we have an edge in G from $\widehat{x}$ to $\widehat{y}_1$. We have the edge $\widehat{x}\widehat{y}$ in H. The proofs are similar for the three other cases (x in Case 2, y in Case 2 etc.).

Conversely, assume that we have an edge $\widehat{x}\widehat{y}$ in H. Then, we have $x = \rho_u(uv)$ and $y = \rho_u(uw)$ with $v \neq w$. There are four cases, e.g., x is in Case 1 and y in Case 2. Then, we get easily that xy is an edge of H_u by considering alternating paths. □

This completes the proof of the proposition because $cwd(G') \leq cwd(G)$.□

Remarks 5.16 : (1) By the recent result in [17], we can use the mapping $f(k) =: k^2.3^{k-1}$ to bound the clique-widths of components.

(2) The proof of Proposition 5.14 given in [8] (as Proposition 4.16) is incorrect: Lemma A.2.3 shows (correctly) that $cwd(H) \leq 4cwd(G)$ if H is obtained from G by fusing two vertices. But, in order to prove the statement, one must fuse the vertices of several pairs (as we do above to define H from G'), hence, one does not obtain a bounding function f as claimed. □

An alternative split decomposition for directed graphs.

Kanté and Rao have defined in [33] the *displit decomposition* of a directed graph. For an undirected graph, it is the same as the split decomposition. It is incomparable to the split decomposition of [18] because the prime components are different. However, every connected directed graph has a unique decomposition.

Furthermore, for an appropriate notion of rank-width for directed graphs [32], they obtain that the rank-width of a graph is the least upper-bound of the rank-widths of the components of its displit decomposition (cf. Remark 4.16). They also characterize the directed graphs of rank-width at most 1 in a way that generalizes the various characterizations of distance-hereditary graphs (in particular that of [35]).

We think that the results of this section and of [8] (the existence of monadic second-order transformations between directed graphs and their canonical split decompositions) can be extended to displit decompositions.

6 Conclusion

Our purpose was to clarify the relations between split-decompositions for directed and undirected graphs, substitutions and the related graph grammars, and to obtain good bounds on the clique-widths of the defined graphs. For doing that we had to generalize the notion of substitution used in the theory of modular decomposition (cf. Definitions 2.1 and 5.1).

Our open questions concern bounds on the clique-widths of graphs defined by edge-contractions : see Lemma 1.3, Theorem 4.15 and Theorem 5.13. (Results will appear in [17]).

References

[1] H.-J. Bandelt and H. Mulder, Distance-hereditary graphs, *Journal of Combinatorial Theory, Series B*, **41** (1986) 182–208.

[2] H. Bodlaender and A. Koster, Treewidth computations I. Upper bounds. *Inf. Comput.* **208** (2010) 259-275.

[3] T. Bouvier, *Graphes et décompositions*, Doctoral dissertation, Bordeaux University, 2014.

[4] A. Brandstädt and V.B. Le, Structure and linear time recognition of 3-leaf powers. *Inf. Process. Lett.* **98** (2006) 133-138.

[5] M.S. Chang, S.Y. Hsieh and G.H. Chen, Dynamic programming on distance-hereditary graphs, *Proceedings of ISAAC 1997, Algorithms and Computation*, *Lec. Notes Comput. Sci* 1350, Springer, 1997, 344-353.

[6] S. Cicerone and G. Di Stefano, On the extension of bipartite to parity graphs. *Discrete Applied Mathematics* **95** (1999) 181-195.

[7] B. Courcelle, An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theor. Comput. Sci.* **55** (1987) 141-181.

[8] B. Courcelle, The monadic second-order logic of graphs XVI : Canonical graph decompositions. *Logical Methods in Computer Science* **2** (2006).

[9] B. Courcelle, On the model-checking of monadic second-order formulas with edge set quantifications, *Discrete Applied Mathematics* **160** (2012) 866-887.

[10] B. Courcelle, Clique-width and edge contraction. *Inf. Process. Lett.* **114** (2014) 42-44.

[11] B. Courcelle, From tree decompositions to clique-width terms, https://hal.archives-ouvertes.fr/hal-01398972, *Discrete Applied Mathematics*, 2018, to appear, https://doi.org/10.1016/j.dam.2017.04.040.

[12] B. Courcelle and I. Durand, Automata for the verification of monadic second-order graph properties, *J. Applied Logic* **10** (2012) 368-409.

[13] B. Courcelle and I. Durand, Computations by fly-automata beyond monadic second-order logic, http://hal.archives-ouvertes.fr/hal-00828211, *Theor. Comput. Sci*, **619** (2016) 32-67. Short version in *Proc. Conference on Algebraic Informatics*, *Lecture Notes in Computer Science* **8080** (2013) 211-222.

[14] B. Courcelle and J. Engelfriet, *Graph structure and monadic second-order logic, a language theoretic approach*, Volume **138** of *Encyclopedia of mathematics and its application*, Cambridge University Press, June 2012.

[15] B. Courcelle, P. Heggernes, D. Meister, C. Papadopoulos and U. Rotics, A characterisation of clique-width through nested partitions, *Discrete Applied Maths*, **187** (2015) 70-81.

[16] B. Courcelle, J. Makowsky and U. Rotics, Linear-time solvable optimization problems on graphs of bounded clique-width, *Theory Comput. Syst.* **33** (2000) 125-150.

[17] B. Courcelle and M. Raskin, Article in preparation, 2018.

[18] W. Cunningham, Decomposition of directed graphs, *SIAM. J. on Algebraic and Discrete Methods*, **3** (1981) 214–228.

[19] R. Diestel, *Graph theory*, Springer, 2006.

[20] R. Downey and M. Fellows, *Fundamentals of parameterized complexity*, Springer-Verlag, 2013.

[21] I. Durand, TRAG: Term Rewriting Automata and Graphs, software developped since 2015, http://dept-info.labri.u-bordeaux.fr/~idurand/trag

[22] I.Durand and M.Raskin, TRAG-WEB: Term Rewriting Automata and Graphs (online), Web interface under development, 2018, https://trag.labri.fr

[23] M. Fellows, F. Rosamond, U. Rotics and S. Szeider, Clique-width is NP-complete. *SIAM J. Discrete Math.* **23** (2009) 909-939.

[24] E. Fischer J. Makowsky and E. Ravve, Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics* **156** (2008) 511-529.

[25] E.Gioan and C. Paul, Split decomposition and graph-labelled trees: Characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Applied Mathematics* **160** (2012) 708-733.

[26] E. Gioan, C. Paul, M. Tedder and D. Corneil, Practical and efficient split decomposition via graph-labelled trees. *Algorithmica* **69**(2014): 789-843.

[27] M. Golumbic and U. Rotics: On the Clique-Width of Some Perfect Graph Classes. *Int. J. Found. Comput. Sci.* **11** (2000) 423-443.

[28] F. Gurski, The behavior of clique-width under graph operations and graph transformations. *Theory Comput. Syst.* **60** (2017) 346-376.

[29] M. Habib and C. Paul, A survey of the algorithmic aspects of modular decomposition. *Computer Science Review* **4** (2010) 41-59.

[30] P. Heggernes, D. Meister and C. Papadopoulos, Characterising the linear clique-width of a class of graphs by forbidden induced subgraphs, *Discrete Applied Mathematics* **160** (2012) 888-90.

[31] P. Hlinený, S. Oum, D. Seese and G. Gottlob, Width parameters beyond tree-width and their applications. Comput. J. **51** (2008) 326-362.

[32] M. Kanté and M. Rao, The rank-width of edge-coloured graphs. *Theory Comput. Syst.* **52** (2013) 599-644.

[33] M. Kanté and M. Rao, Directed rank-width and displit decomposition. Proceedings of WG 2009, pp. 214-225.

[34] D.Meister, Clique-width with an inactive label. *Discrete Mathematics* **337** (2014) 34-64.

[35] S. Oum, Rank-width and vertex-minors, *Journal of Combinatorial Theory, Series B*, **95** (2005) 79–100.